ioProgrammo Anno IX - N°5 (91)

PER ESPERTI E PRINCIPIANTI

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • MAGGIO 2005 • ANNO IX, N.5 (91)

LA GRAFICA
COSTRUITA INTORNO A TE

IMPARA XUL, IL LINGUAGGIO FACILE NATO PER REALIZZARE INTERFACCE A MISURA D'UOMO

- Personalizza FireFox costruendo, da solo, i suoi plugin
- Usa i fogli di stile per posizionare pulsanti e immagini
- Gestisci gli eventi semplicemente con JavaScript

DISEGNARE CON PHP
Nel browser un universo di immagini, linee, forme, punti e colori a cui dare vita

VISUAL BASIC CONTROLLO TOTALE
Accedi direttamente alla memoria fisica e leggi le informazioni che il sistema ti nasconde

JAVA

COSTRUIRE UN PLAYER MP3
Il vantaggio di un linguaggio cross platform per lo sviluppo di applicazioni multimediali

BLUETOOTH SECURITY
Utilizzare il numero seriale del cellulare come una chiave d'accesso hardware

IO PROGRAMMO WEB

ERRORE 404 MIO!
Personalizzare le pagine non trovate, per dare più feedback all'utente

C# & VB.NET

REMOTING
La tecnica usata in .Net per sviluppare applicazioni che comunicano fra loro

ACCESS AL VOLO
Creare un database e usarlo senza disporre del programma

WATERMARK DI IMMAGINI
Stop ai download facili. Applichiamo un marchio alle nostre creazioni

C++

SIMPLE DIRECTMEDIA LAYER
Potenza senza limiti! Accesso a basso livello ad audio, video e tastiera

JAVA

FILTRI PER L'ELIMINAZIONE DEL RUMORE
Addio disturbi su immagini, video e suoni, con il Digital Signal Processing

JBOSS
Facciamo conoscenza con l'application server principe per l'enterprise business

I CORSI PER IMPARARE
• VISUAL BASIC.NET 2003
• SYMBIAN • ASP.NET

Estendere FireFox con i plugin • Visual Basic controllo totale • Disegnare con PHP • Gestire la sicurezza con BlueTooth

KERNEL: I SISTEMI OPERATIVI MODERNI E IL MULTITASKING

EDIZIONI MASTER

Una questione di libertà

In questo numero di ioProgrammo pubblichiamo il reportage dell'intervento di Richard Stallman tenuto in Italia in occasione di un convegno organizzato da AssoEtica. Il contenuto dell'intervento di Stallman è assolutamente semplice ed intuitivo da comprendere, espone le libertà fondamentali che hanno fatto dell'OpenSource un movimento così rivoluzionario nel mondo, ma è facile riassumere l'intero Stallman Pensiero in poche ma fondamentali parole: "Il software è un bene che aiuta l'umanità ad evolvere e tutte le cose che vanno in questa direzione devono essere fruibili a chiunque abbia la capacità di usarle e migliorarle". Ora, qui non vogliamo porre l'accento su una guerra di religione sulle varie modalità di intendere il business intorno al software, vogliamo però indicare una direzione, esporre il nostro punto di vista da persone che fanno formazione se pur a mezzo della carta stampata. E' importante che le aziende spostino il loro focus dalla vendita del software all'assistenza, alla personalizzazione ed ai servizi, è forse una via più difficile ma di gran lunga più redditizia, inoltre consente di fidelizzare i propri utenti. La maggior parte delle grandi aziende sta utilizzando questo modello, ed è ora che anche le piccole software house, i singoli programmatori inizino a pensare a un modo diverso di intendere la "vendita" di un pacchetto software, ovvero come la vendita di un servizio utile alla crescita complessiva delle aziende che lo utilizzano, pertanto capace di adattarsi a chi lo usa, supportato e sufficientemente aperto a modifiche e cambiamenti.

Fabio Farnesi

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **gatto** *Password:* **matita**

LA GRAFICA COSTRUITA INTORNO A TE

Impara XUL il linguaggio facile nato per costruire interfacce a misura d'uomo

- **Facile come XML**
- **Pochi tag, semplici ma potenti**
- **Gestisce gli eventi in modo intuitivo**

pag. 16

Anno IX - N.ro 5 (91) - Maggio 2005 - Periodicità Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Raffaele del Monaco, Fabio Farnesi
Collaboratori: M. Autiero, M. Bigatti, D. Bochicchio, L. Buono, F. Grimaldi, D. Fadda, F. C. Ferracchiati, A. Marroccelli, S. Meschini, V. Muraglia, G. Natili, F. Paparoni, P. Perrotta, M. Poponi, F. Smelzo, L. Spuntoni, A. Trapani, I. Venuti, F. Vaccaro.
Segreteria di Redazione Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale: SET S.r.l.
Coordinamento Tecnico: Piero Mannelli
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo Basic (11 numeri) €52,90 sconto 30% sul prezzo di copertina di €75,90 ioProgrammo con Libro (11 numeri + libro) €76,00 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 30/04/05
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Ariberto, 24 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Ariberto, 24 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore - zona ASI Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Aprile 2005

Edizioni Master edita: Computer Bild, Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan, Digital Music, Horror Mania, ioProgrammo Extra, Le Collection.

SICUREZZA BLUETOOTH

Realizziamo un modulo per proteggere le nostre applicazioni senza bisogno di password pag. 66

http://forum.ioprogrammo.it

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

News

ESTRADIZIONE PER GLI HACKER

Tempi duri per i pirati. Fra arresti e sanzioni pecuniarie ecco che arriva anche la prima estradizione. Raymond Griffiths quarantaduenne appartenente alla Crew di DrinkOrDie nota anche come DoD, è accusato dagli stati uniti di avere violato i diritti sul copyright del software relativamente a centinaia di opere. L'operazione contro Griffiths è da inquadrare in quella più generale contro DoD che nel corso degli anni avrebbe diffuso software "pirata" per un valore complessivo di cinquanta milioni di dollari. La richiesta di estradizione è singolare, Griffiths è infatti australiano ed ha sempre vissuto in Australia, per contro l'accusa gli è stata mossa dagli Stati Uniti. Vedremo come andrà a finire, certo è che Hacker, Pirati e quanti utilizzano la rete in modo illegittimo hanno poco da stare allegri, ed era ora… aggiungiamo noi.

FIREFOX SI AGGIORNA

Uno dei cavalli di battaglia usati da Firefox contro Internet Explorer è proprio la presunta scarsa sicurezza del secondo. Fino ad ora Microsoft e gli analisti avevano obiettato che solitamente il prodotto più diffuso è anche quello maggiormente attaccato, pertanto ovviamente Explorer doveva fare i conti più spesso di Firefox con la pericolosità degli Hacker. Se così fosse bisogna dire che Firefox comincia a essere diffuso e paradossalmente subisce i primi attacchi, di fatto è stata appena rilasciata una security patch che lo porta alla versione 1.0.2 e colma alcune lacune proprio sulla sicurezza.

LA MELA SENZA UN TASTO

Potrebbe cadere l'ultimo dogma di Apple. Per ora è solo una notizia non confermata, un rumors di quelli che popolano i blog di internet e si diffondono senza nessuna precisa conferma, ma circola la voce che Apple potrebbe mettere a disposizione dei suoi utenti un mouse a due tasti, tipico del mondo dei PC. Il costo si aggirerebbe intorno ai € 60.

Potrebbe trattarsi di una rivoluzione per gli utenti Apple, che dopo essersi ritrovati con un sistema di derivazione Unix sotto la mela adesso potrebbero addirittura approdare ai due tasti del PC.

I PROBLEMI PER L'ECOMMERCE

Con il comma 7 del primo articolo del Decreto Legge del 14 Marzo 2005,, viene punito con una multa fino a 10.000 euro chi non si accerta della provenienza di un bene prima di acquistarlo o chi favorisce lo scambio di beni senza averne accettato la legittima provenienza. E fin qui tutto sembra equo e sottoscrivibile, se non che l'intero mondo delle Aste OnLine eBay intesta si è vista improvvisamente crollare il mondo in testa. Come lo spinoso problema possa essere risolto è ancora da scoprire.

I CINQUE COLPI DI LINUX

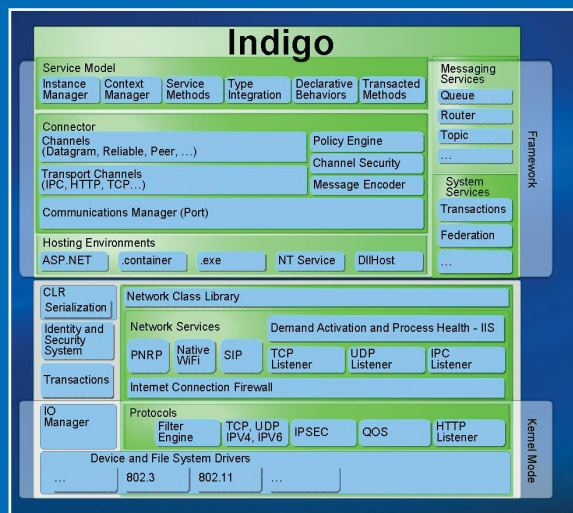
Kde 3.4, Gnome 2.10, OpenOffice 2.0, Acrobat Reader 7 e, incredibile ma vero, una versione per Linux del famoso software di masterizzazione "Nero Burning Rom", sono questi i cinque colpi messi a segno in un solo mese dal temibile pinguino. Si tratta veramente di cinque rilasci importanti.
Gnome 2.10 segna l'avvento di un'interfaccia rivoluzionaria, degna sicuramente di quella che gira nelle varie piattaforme Windows e al di sotto solo di pochi scalini rispetto a Aqua, l'interfaccia MacOS X che viene considerata, al momento, la migliore disponibile sul mercato.
D'altra parte il rivale Kde 3.4 non sta a guardare e introduce una miriade di novità, compreso persino il supporto vocale alla gestione del desktop. Sul fronte delle applicazioni si segnala il rilascio dell'attesissimo OpenOffice 2.0, la suite per l'ufficio corrispondente alla famosa e utilizzatissima suite Microsoft Office. OpenOffice 2.0 introduce una serie sterminata di novità, ma spicca fra le altre l'attesa presenza di un'applicazione di gestione dei database che si contrappone al noto Access presente nelle piattaforme Microsoft. Il database gestito da questa piattaforma sarà HSQL DB. Le ultime due novità non riguardano direttamente il sistema ma si tratta del porting di due

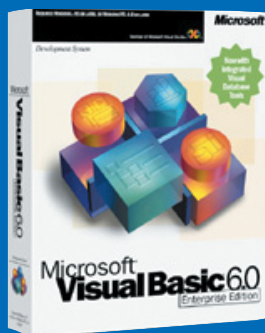
SUN RILASCIA UN SECONDO UPGRADE PER JAVA 5.0

Non ci sono novità strutturali per questo secondo upgrade, tuttavia vengono fissati numerosi Bug, nell'ordine delle centinaia. Molti Bug Fixes riguardano il compilatore ma la stragrande maggioranza fissa dei bug contenuti nei vari package. Spiccano ovviamente i bug fixes relativi ai pacchetti Swing, Awt e 2D che per la loro diffusione rappresentano il cuore pulsante di Java. Per conoscere la versione del JDK che avete installato, potete digitare: *java -version*

applicazioni molto utilizzate in ambiente Windows. Per quello che concerne Acrobat Reader, in realtà esisteva già una versione Linux, ovvero la 5.0, bisogna però dire che questa versione 7 è assolutamente perfetta, la dove la 5 lasciava ancora a desiderare in ambiente Linux in termini di interfaccia e facilità d'uso.
Infine, ultima nota, ma non meno importante delle altre, anzi per certi versi rappresenta forse la novità più interessante: Nero Burning Rom girerà su piattaforma Linux. Se da un lato questo assume un'importanza rilevante a causa della posizione di assoluto dominio che Nero ha fra i software di masterizzazione, dall'altro diventa ancora più importante perchè si registra la volontà di una software house primaria di investire fortemente nel segmento Linux.

WINDOWS LUCE VERDE PER INDIGO

Microsoft ha rilasciato una *Community Test Preview* di Indigo. Lanuova versione è stata rilasciata agli sviluppatori per mettere in mostra le nuove funzionalità dell'architettura di Microsoft dedicata al software basato su Web Services. La community test preview è stata resa disponibili a tutti gli utenti con abbonamento MSDN. Microsoft sembra mostrare una particolare attenzione sia verso Indigo che verso Avalon ovvero l'interfaccia grafica che assieme al nuovo file system WinFS supporterà Longhorn, il nuovo sistema operativo di Microsoft che vedrà la luce nel 2006. È interessante notare come l'architettura dei Web Services che almeno in Italia non ha ottenuto un successo eclatante, continui ad essere invece al centro sia degli strumenti di sviluppo di Microsoft che dell'intera strategia alla base dei nuovi sistemi operativi.

MICROSOFT NON SUPPORTERÀ PIÙ VB6

**Già nel 1998 il gigante di Redmond aveva annunciato la sua volontà di cessare il supporto a Visual Basic 6 e già nel 1998 l'intera comunità degli sviluppatori aveva espresso la propria perplessità e molte aziende avevano deciso di migrare ad altri sistemi di sviluppo.
L'evidente difficoltà della piattaforma .NET a rimpiazzare il suo predecessore deve avere convinto Microsoft a rinnovare le proprie intenzioni. In un'intervista rilasciata a news.com Soma Somasegar vicepresidente della Microsoft's tools division ha dichiarato che l'intenzione di Microsoft è supportare gli utenti nel passaggio verso la nuova piattaforma .NET.
Per fare questo non solo saranno introdotte in Visual Studio funzionalità che aiutino alla migrazione, ma un'intera area del sito di Microsoft sarà dedicata al supporto alla migrazione.
Intanto in alcuni messaggi provocatori comparsi su Internet, le community degli sviluppatori propongono di dare il vita a un clone freeware di VB6 e continuare a utilizzare il loro strumento di sviluppo preferito. E c'è già chi pensa a una petizione online.**

ARRIVA INTERNET EXPLORER 7

Microsoft non poteva restare immobile a subire gli attacchi dell'agguerrita concorrenza. Opera prima Firefox dopo hanno scagliato un attacco senza precedenti al leader incontrastato del mercato, minandone alle basi la credibilità e proponendo delle ottime soluzioni alternative. È il momento della risposta. Per il momento sono ancora rumors, non c'è una data di rilascio certa e le notizie sono frammentarie. Sembrerebbe che il nome in codice del progetto sia "Rincon". Lo sforzo maggiore nel produrre questa nuova versione sarebbe stato diretto a migliorare le politiche di sicurezza. Dal punto di vista dell'usabilità dovrebbero fare la loro comparsa le tanto attese "tabsheet" che hanno fatto la fortuna sia di Firefox che di Opera. Le ultime due novità riguarderebbero il supporto al formato immagine PNG e la presenza di un news aggregator. Il rilascio di una prima beta di Internet Explorer 7 potrebbe essere effettuato durante l'estate.

INBox
L'esperto risponde...

Cosa vuol dire NAT?

Gentile redazione di ioProgrammo, dopo lunghe peripezie ed essendomi ingegnato parecchio per capirci qualcosa, anche grazie ad un suggerimento trovato nella rubrica della posta di ioProgrammo di qualche numero fa, sono riuscito a configurare la mia piccola rete casalinga. Ho tre pc, di cui uno portatile e un router. Adesso vorrei fare in modo di pubblicare un mio sito web utilizzando la mia ADSL di casa. Non ho ben chiaro i passi da compiere, mi hanno detto che ho bisogno del NAT, che cosa è? Come funziona?

Emanuele da forlì

Gentile Emanuele. Il problema non è complesso ma richiede una serie di piccoli passi che necessitano di essere spiegati nel dettaglio.
Partiamo proprio dal NAT e dalla configurazione della rete interna. Se hai configurato la rete interna, devi averlo fatto utilizzando un indirizzo del tipo 192.168.0.x per ciascun computer della rete, o 10.10.10.x insomma un indirizzo di classe protetta. Fatto questo i tuoi computer riescono ugualmente a navigare perché il router /gateway fa da ponte fra gli indirizzi della tua rete interna e l'unico indirizzo di rete esterna di cui la tua ADSL è dotata.
Potresti avere anche più di un indirizzo esterno ma la maggior parte delle ADSL casalinghe vengono vendute con un solo IP esterno disponibile quindi ci atterremo a questa configurazione.
Come ben saprai gli indirizzi di rete interna non sono visibili da internet, va da se che un client dall'altra parte del globo scrivesse *http://192 .168.0.1* non accederebbe affatto a una delle macchine della tua rete interna ma ad una macchina locale alla rete interna del client.
Viceversa digitando *http://indirizzodeltuorouter* riceverebbe un errore perché molto probabilmente sul tuo router non c'è un server web sulla porta 80, inoltre presumibilmente il tuo server web girerà su una macchina interna alla tua lan e non sul router.
Pertanto il NAT è proprio quel servizio che si occupa di fare passare i pacchetti dall'esterno della rete verso la tua lan. In sostanza richiamando *http://indirizzodeltuorouter* il NAT individuerà che è stata fatta una richiesta alla porta 80, controllerà una tabella di associazione e se nella tabella scoprirà che una delle macchine della tua lan è configurata come server web indirizzerà i pacchetti dalla rete esterna verso quella interna così che la richiesta possa oltrepassare la barriera del router.
Ovviamente il Nat deve essere configurato sul router, e ciascun router prevede istruzioni in tal senso, è il caso di controllare il manuale allegato al tuo.
Il secondo passo è correlato alla registrazione di un dominio Internet. Registrare un dominio significa associare un nome a un indirizzo IP. Affinché il tuo server web possa essere acceduto come di consueto digitando il nome nel browser è necessario che l'IP del tuo router sia associato ad un dominio internet. Questa operazione deve farla un provider che deve provvedere anche ad associare il nome al tuo IP tramite un servizio di DNS.
Ovviamente per poter fare questa operazione devi avere un ADSL che preveda un IP statico e non dinamico come la maggior parte delle ADSL in commercio. Se così non fosse potresti usare un servizio come DynDNS – *http://www.dyndns.org* – questo tipo di servizio tramite un apposito programma aggiorna dinamicamente un dns rendendo sempre aggiornata l'associazione del nome all'IP. Tuttavia c'è da dire che il NAT potrebbe avere qualche problema con questo tipo di configurazione. Compiute queste operazioni non resta che installare un server web, Apache o IIS su una delle macchine della rete interna configurata sul NAT e tutto funzionerà come di consueto.
Ovviamente qui abbiamo riportato un esempio di configurazione con un server http , in realtà il NAT funziona allo stesso modo con tutti i protocolli. Sostanzialmente fa da ponte tra le richieste che provengono dall'esterno e la tua rete interna.

Impedire il download diretto di un file

Mitici programmatori dalle dita consumate. Chiedo a voi lumi sul seguente problema: sto sviluppando un sito in PHP. A un certo punto desidero che solo gli utenti registrati possano scaricare alcuni file. Non ho problemi per gestire l'autenticazione. Il punto è che se qualcuno prova direttamente a caricare il percorso http://www.miosito .ext/filename.zip riuscirà lo stesso a scaricare il file pur non avendone i permessi. Come posso aggirare il problema?

Luigi da Asti

Luigi, il tuo problema non è così complesso come sembra. Prima di tutto non devi mettere i file che vuoi proteggere in una directory visi-

bile sul web. Potresti anche farlo ma questo implicherebbe che il meccanismo di protezione sia da imputare al server web e non a PHP, è una soluzione valida ma in tal caso gli utenti registrati del tuo sito in un qualche modo dovrebbero anche essere utenti del sistema e questo non è consigliabile. Una soluzione più interessante è quella di mettere i file in questione in una directory non visibile sul web.

A questo punto se l'utente supera la fase di autenticazione puoi spedirgli il file tramite una funzione Header, l'esempio è il seguente:

```
header( 'Content-Type: application/zip );
header( 'Content-Size: $fileSize );
header( "Content-Disposition: attachment;
                        filename=\"$fileName\"");
@readfile($file);
```

Ovviamente preoccupati di riempire le variabili con i nomi e i percorsi corretti. Inoltre dovresti controllare che il Content-Type corrisponda a quello del file che desideri sia downloadato.

Compilatori Intel

Gentile redazione di ioProgrammo. Ho sentito recentemente parlare di alcuni compilatori C++ prodotti da Intel. Non ho ben capito quando e perché è conveniente usarli, ed esattamente che caratteristiche hanno. Potreste darmi qualche delucidazione?

Alberto da Trapani

Buongiorno Alberto. Intel è una società che ha interessi piuttosto ramificati in quasi ogni settore dell'Information Technology. In questo senso non poteva mancare una divisione esclusivamente rivolta ai programmatori. In particolare una pagina piuttosto articolata che mette in luce tutte le proposte che Intel rivolge agli sviluppatori è raggiungibile all'indirizzo: *http://www.intel.com/software/products/index.htm*.

Per quanto riguarda le caratteristiche dei compilatori in questione, il loro maggior pregio è quello di essere particolarmente performanti. Vengono ottimizzati sulla base dei processori, per cui sono particolarmente indicati in tutte quelle applicazioni dove la velocità è fondamentale, vedi ad esempio il real time.

Esistono versioni sia per Windows che per Linux, inoltre è importante sottolineare che i compilatori Intel esistono non solo per il linguaggio C++ ma anche per il linguaggio Fortran. In tutti i casi sono pronti per l'architettura a 64 bit ed è possibile sviluppare con questi compilatori utilizzando Eclipse. Infine è opportuno segnalare che esistono tool specifici per misurare proprio le performance dei software ottenuti con questi compilatori. Si tratta di ottimi prodotti, senza dubbio destinati a un mercato professionale dove certi meccanismi di tuning sono essenziali. In ogni caso direttamente dal sito di Intel è possibile downloadare le versioni Trial dei prodotti in questione all'indirizzo *http://www.intel.com/software/products/global/eval.htm*. Chi li ha provati garantisce che non tornerebbe mai indietro.

A chi spedire la posta?

Alla nostra redazione arriva spesso un considerevole numero di email riguardante temi specifici. Per consentirci di rispondere velocemente e in modo adeguato alle vostre domande abbiamo elaborato una FAQ – Frequently Ask Question – o risposte alle domande frequenti in italiano, di modo che possiate indirizzare le vostre richieste in modo mirato.

Problemi sugli abbonamenti
Se la tua domanda ha a che fare con una delle seguenti:

- Vorrei abbonarmi alla rivista, che devo fare?
- Sono un abbonato e non ho ricevuto la rivista, a chi devo rivolgermi?
- Sono abbonato ma la posta non mi consegna regolarmente la rivista, a chi devo rivolgermi?

Contatta **abbonamenti@edmaster.it** specificando che sei interessato a ioProgrammo. Lascia il tuo indirizzo email e indica il numero dal quale vorresti far partire l'abbonamento. Verrai contattato al più presto. Oppure puoi chiamare lo **02 831212**

Problemi sugli allegati
Se riscontri un problema del tipo:

- Ho acquistato ioProgrammo ed il Cdrom al suo interno non funziona. Chi me lo sostituisce?
- Ho acquistato ioProgrammo ma non ho trovato il cd/dvd all'interno, come posso ottenerlo?
- Vorrei avere alcuni arretrati di ioProgrammo come faccio?

Contatta **servizioclienti@edmaster.it**
Non dimenticare di specificare il numero di copertina di ioProgrammo e la versione: con libro o senza libro. Oppure telefona allo **02 831212**

Assistenza tecnica
Se il tuo problema è un problema di programmazione del tipo:

- Come faccio a mandare una mail da PHP?
- Come si instanzia una variabile in c++?
- Come faccio a creare una pagina ASP.NET

o un qualunque altro tipo di problema relativo a tecniche di programmazione, esplicita la tua domanda sul nostro forum: **http://forum.ioprogrammo.it**, uno dei nostri esperti ti risponderà. Le domande più interessanti saranno anche pubblicate in questa rubrica.

Problemi sul codice all'interno del CD
Se la tua domanda è la seguente

- Non ho trovato il codice relativo all'articolo all'interno del cd

Consulta la nostra sezione download all'indirizzo **http://cdrom.ioprogrammo.it**, nei rari casi in cui il codice collegato ad un articolo non sia presente nel cdrom, senza dubbio verrà reso disponibile sul nostro sito.

Idee e suggerimenti
Se sei un programmatore esperto e vuoi proporti come articolista per ioProgrammo, oppure se hai suggerimenti su come migliorare la rivista, se vuoi inviarci un trucco suggerendolo per la rubrica tips & tricks invia una email a **ioprogrammo@edmaster.it**

NEWSGROUP
Le informazioni nella Rete

Direttamente dal forum di ioProgrammo le discussioni più "Hot" del momento

Visual Basic

Scrivere su file .txt i contenuti di molte textbox

Avrei necessita di scrivere il contenuto di molte text box su un file di testo e poi eventualmente di ricaricarlo non so davvero da che parte iniziare mi potete gentilmente dare qualche idea.
Un grazie anticipatamente.

shadow666

http://forum.ioprogrammo.net/thread.php?threadid=5173&boardid=13

Risponde iochico

Supponendo di avere in un form 3 textbox *(text1, text2, text3)* nelle quali sono contenute, rispettivamente, la parola *"Pippo", "Pluto", "Paperino"*, puoi ottenere quanto richiesto, utilizzando questo codice:

```
Option Explicit
Private Sub command1_click()
Dim control As control, stringa As String
Open "C:/miofile.txt" For Output As #1
For Each control In Me.Controls
If TypeOf control Is TextBox Then
stringa = "[" & control.Name & "]"
                    & control.Text
Print #1, stringa
End If
Next
Close #1
End Sub
```

alla fine nel file *miofile.txt* troverai il seguente risultato:

```
[Text3] Pippo
[Text2] Pluto
[Text1] Paperino
```

Per la lettura dei dati esegui l'operazione inversa....

Aprire una pagina Web da VB.NET

Sto usando vb.net per un applicazione webform.ora come posso da codice aprire un'altra pagina ???.
Adesso ci riesco solo con l'oggetto hyperlink.

totti240282

http://forum.ioprogrammo.net/thread.php?threadid=5144&boardid=27

Risponde gekki

Metti un bottone sulla webform e utilizza il seguente codice:

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System
        .EventArgs) Handles Button1.Click
Response.Redirect("http://www.google.it")
End Sub
```

PHP

Popolamento dinamico tabella...

Vorrei far in modo che una tabella (aggiunta ad una pagina web tramite tag *<table>*...) venga popolata dinamicamente con i nomi dei file presenti in una determinata directory.
Ad Esempio: mettiamo che nella dir ci siano i file *1.txt 2.txt* e *3.txt;*
la tabella dovrà mostrare:

1
2
3

Ovviamante linkati......
La dinamicità sta nel fatto che aggiornando i dati nella directory, automaticamente lo script esegue un refresh sulla tabella aggiornandola.....in poche parole legge tutto quello che si trova all'interno della dir. Come posso fare?

Belva

http://forum.ioprogrammo.net/thread.php?threadid=5164&boardid=2

Risponde Jachetto

Ecco a te un esempio di codice che potrebbe esserti utile.

```
$directory="./";
$open_handle=opendir($directory);
while (($read=readdir($open_handle) and
                        is_dir($read))) {
    $open_sub=opendir($read);
    while ($sub_read=readdir(
                        $open_sub)) {
        echo "<tr><td>".$sub_read."&
                        </td></tr>";
    }
}
```

In realtà questo codice controlla aggiunge un ulteriore elemento a quanto da te richiesto, controlla cioè nelle directory immediatamente sotto quella di partenza. In ogni caso partendo da questo esempio puoi personalizzare quanto vuoi. Ho lasciato anche aperta la possibilità di gestire le directory "." e ".." devi solo aggiungere un *if*.

Java instanziare una variabile

Salve, ho cominciato da poco a programmare in Java. So che la mia è una domanda banale, ma sembra proprio che io non riesca a venirne a capo.

Chiedo a voi tutti un suggerimento per istanziare una variabile di tipo int che ma che non sia un array per intenderci non qualcosa come:

```
...
int[] vett = new int[5];
...
```

ma qualcosa come

```
...
int num = new int;
...
```

devo comunque ammettere che non so se ciò che ho chiesto sia qualcosa di realmente possibile.

il piccolo Alex

http://forum.ioprogrammo.net/thread.php?threadid=5130&boardid=18

Risponde Nexol

Ciao piccolo Alex.
Vuoi definire una variabile di tipo intero? Detto, fatto:

```
int num;
```

Vuoi definire una variabile di tipo intero e contemporaneamente assegnargli un valore, che so, 5? Detto, fatto:

```
int num = 5;
```

Vuoi soltanto assegnare il valore 5 alla variabile num che hai già precedentemente creato? Detto, fatto:

```
num = 5;
```

C++

Vettori troppo grandi

Ho notato che in c/c++ non vi è un controllo automatico sulla dimensione massima di un vettore. Ad esempio il seguente codice:

```
int main (void) {
    double vett[10000000];
    vett[0] = 1.;
    return 0;
}
```

mi da segmentation fault. Qualcuno conosce un modo per controllare tali inconvenienti?

Frea

http://forum.ioprogrammo.net/thread.php?threadid=5152&boardid=20

Risponde johnkoenig

Prova con un codice simile:

```
#include "iostream.h"
int main(int argc, char* argv[])
{
   double * vett;
    int dimensione = 100000000;
   if ((vett = new double[dimensione]) != NULL)
   {
      vett[0] = 1.;
     cout<<"Valore= "<<vett[0]<<"\n\n";
   }
else
{
   cout<<"\n\nMemoria insufficiente...\n\n";
}
if (vett != NULL)
delete vett;
return 0;
}
```

Il limite consiste solo nella massima area di memoria dimensionabile (tenendo conto che in genere si lavora su sistemi a 32bit = 4294967296 byte = 4.096 Mbyte.
Naturalmente, il numero degli elementi dipende dal tipo di dato con cui si lavora.

Macromedia Flash

aaaa

Ho da poco imparato ad utilizzare flash mx 2004 professional con il relativo action script. Ho la necessità di dover creare a tempo di esecuzione una serie di textField in base al contenuto letto in file XML, ho provato ad utilizzare un algoritmo del genere ma non mi viene visualizzato nulla.

```
var nometxt;

for (i=0; ..nodi xml...; i++) {
nometxt = "nome"+i+"_txt";
_root.createTextField(nometxt, 1, 250, 100, 200, 20);
nometxt.border = true;
nometxt.text = "...valore di un nodo xml..."
};
```

Più in generale, come posso creare attraverso un ciclo una serie di textField? Grazie in anticipo per le vostre risposte.

devnic

http://forum.ioprogrammo.net/thread.php?threadid=4882&boardid=38

Risponde xxxxx

Devi fare due modifiche:

1) la depth, il secondo argomento, non deve essere sempre uguale, quindi utilizza la variabile i

2) per popolare e lavorare sul campo usa o eval() o il this con gli operatori di accesso all'array

```
eval(nometxt).text = ....;

_root["nome"+i+"_txt"].text = ....;
```

Risponde Giorgio Natili

```
for (i=0; i<nodiXml; i++) {
    nometxt = "nome"+i+"_txt";
    var tmpClip = attachMovie("user", nometxt, i);
    tmpClip._x = 380;
    tmpClip._y = 200+(i*60);
     tmpClip.user_txt.text = "Testo da aggiungere"
}
```

SERVIZIO CLIENTI
e-mail: sevizioclienti@edmaster.it
Tel. 02 83 12 12

SOSTITUZIONE CD
Inviare il CD Rom difettoso in busta chiusa a: Edizioni Master Servizio Clienti Via Ariberto, 24 - 20123 (MI)

Una "volpe" per browser

Nel vostro browser o client email preferito manca qualcosa? Basta un buon editor xml per scrivere una estensione che li completi secondo le vostre esigenze

Utilizza questo spazio per le tue annotazioni

Intenet explorer ha un nuovo avversario il suo nome è Firefox. "Volpe di fuoco", un nome da guerra, come il suo fratellino Thunderbird, "Uccello di Tuono", che si propone di sostituire Outlook o per lo meno la sua versione Express. Il browser prodotto da Mozilla deve il suo successo a un'usabilità elevata, una "user experience" che non stravolge chi per anni ha utilizzato il browser Microsoft. E per chi vuole di più, ci sono ampissime possibilità di personalizzazione, grazie a temi ed estensioni, entrambi scritti in Xul, un linguaggio basato su Xml, quindi cross-platform. nato proprio in casa Mozilla per definire le interfacce grafiche.

LA POTENZA DELLA SEMPLICITÀ

XUL è l'acronimo di *XML User-interface Language.* Si tratta di un linguaggio nato per scrivere principalmente interfacce grafiche. In molti lo stanno scegliendo per scrivere interfacce negli ambiti più vari. Se ne prevedono utilizzi in ambienti lontani da Firefox, come in cellulari, palmari e decoder televisivi. Tuttavia rimane anche il linguaggio base per scrivere estensioni per FireFox. Per estensioni si intendono AddIN, piccoli programmi che potenziano le funzionalità di base di FireFox. Il punto di partenza per noi sarà la comprensione del linguaggio. Non ci avventureremo immediatamente nella scrittura di un'estensione, ma cercheremo invece di capire la logica alla base del linguaggio XUL. Possiamo scrivere interfacce grafiche dotate di tutti gli elementi tipicamente utilizzati: bottoni, caselle di testo, toolbars, menu, viste ad albero, checkbox e quant'altro ci possa servire. In più avremo una gestione degli eventi molto accurata, potendo definire persino le "scorciatoie" da tastiera. Il tutto facilmente localizzabile, grazie al supporto alla multilingua. Partiremo da un file Xul da caricare direttamente da remoto. Questo file dovrà essere salvato nella root di un web server e richiamato da firefox tramite il classico *http://www.nomesito.com/nomefile.xul.* Ovviamente il web server dovrà supportare l'estensione xul fra i suoi myme type, alrimenti Firefox cercherà di scaricare il file invece di interpretarlo. Il nostro file .xul conterrà il seguente codice:

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type=
                                        "text/css"?>
<window xmlns="http://www.mozilla.org/keymaster
        /gatekeeper/there.is.only.xul" orient="vertical"
                                    title="Bookmarks">
 <script type="application/x-javascript">
  function loadURL(event) {
    var contentFrame = document.getElementById(
                                     'contentFrame');
    var url = event.target.getAttribute('value');
    if (url) contentFrame.setAttribute('src', url);}
 </script>
 <menubar oncommand="loadURL(event);">
  <menu label="Programmazione">
   <menupopup>
```

PERCHÉ USARE XUL?

Xul può servire per creare interfacce grafiche web alternative al solito html, che potranno, però, essere visibili solo in Firefox. Basterà scrivere un file con estensione ".xul" e renderlo visibile tramite un web server, oppure aprirlo direttamente dal proprio hard disk con Firefox. In alternativa, si può usare per scrivere estensioni per firefox. in questo caso il file deve essere installato in FireFox come estensione. Si tratta di un file .xpi. In realtà nonostante l'estensione è un normale file zip rinominato. L'archivio contiene un file guida per l'installazione e un jar che contiene tutti i file risorsa. In questo articolo introduciamo tutte e due le tecniche, con particolare riferimento alla prima.

```
        <menuitem label="IoProgrammo"   value=
                        "http://www.ioprogrammo.it/" />
        <menuitem label="Javastaff" value=
                        "http://www.javastaff.com/" />
        <menuitem label="XulPlanet"  value=
                        "http://www.xulplanet.com" />
        <menuitem label="Java" value=
                        "http://www.java.sun.com" />
      </menupopup>
    </menu>
    <menu label="Aste">
      <menupopup>
        <menuitem label="Italia"    value=
                        "http://www.ebay.it" />
        <menuitem label="Gran Bretagna"   value=
                        "http://www.ebay.co.uk" />
        <menuitem label="Stati Uniti" value=
                        "http://www.ebay.com" />
        <menuitem label="Francia"  value=
                        "http://www.ebay.fr" />
      </menupopup>
    </menu>
    <menu label="Notizie">
      <menupopup>
        <menuitem label="Ansa"  value=
                        "http://www.ansa.it" />
        <menuitem label="Televideo" value="http://www.
              televideo.rai.it/nazionale/homenaz.asp" />
      </menupopup>
    </menu>
    <button label="WebMail" value=
                        "http://www.gmail.com" />
  </menubar>
  <iframe id="contentFrame" src=
                "http://www.google.it" flex="1" />
</window>
```

Con poche righe di codice riusciamo a creare una barra con dei menu a tendina, completamente funzionante. L'interfaccia grafica è completamente scritta in xul, mentre il controllo è gestito da una piccola funzione in javascript, facilmente integrata grazie al tag *<script>*, che si occupa di aprire la pagina richiesta nel frame al di sotto della barra. La funzione javascript *loadURL(event)* viene richiamata ogni qual volta un utente clicca su uno dei menuitem. Gli elementi fondamentali di questo linguaggio ci sono quasi tutti, a partire dalla dichiarazione del foglio di stile, che si riferisce alla skin impostata nel browser. Notate però che viene utilizzata la URI *chrome://* che va a pescare il foglio di stile nelle risorse interne di FireFox e in particolare nel package global. Utilizzeremo spesso questa URI quando vorremo accedere a risorse interne ai file XPI che gestiscono le estensioni, ma di questo ci occuperemo più avanti. Viene poi definita una *<window>* in cui inseriamo tutti gli elementi grafici da visualizzare.

In una *window* possiamo inserire i più svariati componenti grafici. Qui abbiamo utilizzato menu, di tipo Pop-up, bottoni e un iframe in cui visualizzare la pagina principale. Inserire un menupopup è semplicissimo:

Fig. 1: **Un esempio del codice in azione**

```
<menu label="Notizie">
  <menupopup>
    <menuitem label="Ansa"   value=
                    "http://www.ansa.it" />
    <menuitem label="Televideo"   value=
                    "http://televideo.rai.it" />
  </menupopup>
</menu>
```

La label del menu è l'etichetta che viene visualizzata su ogni singolo menu, i vari *menuitem* vengono visualizzati al click. È l'oggetto *menubar* che indica di eseguire la funzione javascript *loadUrl()* ogni volta che viene intercettato l'evento *oncommand:*

```
  <menubar oncommand="loadURL(event);">
```

La funzione andrà a modificare l'attributo *src* dell'iframe aggiornando quindi la pagina visualizzata.
Il tutto è effettivamente molto semplice e veloce e amplia non di poco le possibilità di un webmaster, permettendogli di disegnare interfacce grafiche che prima necessitavano una interazione fra html e javascript non altrettanto intuitiva.

PIÙ STILE

Firefox interpreta con lo stesso motore sia i file Xul sia HTML e spessissimo i due linguaggi vengono utilizzati contemporaneamente. Per questo è ovvio che i fogli di stile, in formato CSS, sono fondamentali. Vediamo per esempio come sia possibile visualizzare un'immagine in un'interfaccia Xul. Utilizzeremo il componente *<image>*, che si può inserire con una sintassi di questo tipo:

```
<menubar oncommand="loadURL(event);">
  <image src="http://www.google.it
                        /logos/intl_women.gif"/>
```

Questo esempio aggiunge il logo di Google corrente

CONFIGURARE IL WEBSERVER PER USARE XUL

Se vogliamo pubblicare in un webserver dei file xul, dobbiamo configurare correttamente il nostro web server, altrimenti Firefox non riconoscerà il documento e mostrerà semplicemente il sorgente, senza interpretarlo. Nel caso di Apache, dobbiamo modificare il file http.conf inserendo:

```
/AddType
application/vnd
     .mozilla.xul+xml.xul/
```

Invece in IIS dobbiamo seguire i seguenti passi:

1. **Aprire "Strumenti di Amministrazione" nel menu dei programmi;**
2. **Configurazione Server;**
3. **Gestione Servizi IIS;**
4. **Sito Web predefinito -> Proprietà;**
5. **fra le proprietà scegliere quella che permette di configurare i MIME;**
6. **settare */application /vnd.mozilla.xul+xml/* per i file .xul.**

nella menubar. Ovviamente è solo un esempio, non usatelo in questa forma o vi troverete una toolbar gigantesca con il logo di Google in bella evidenza.
La sintassi, che ricorda parecchio l'*<img>* dell'HTML, è corretta, anche se sarebbe meglio, per esempio per supportare più di una skin nella nostra applicazione, utilizzare la seguente:

```
<image id="search"/>
```

E poi nel foglio di stile *css* andare a definire l'id *search:*

```
#search {
   list-style-image: url("chrome:
                      //findfile/skin/images/search.jpg");
 }
```

Abbiamo gia detto che la URI *chrome://* esegue una ricerca sui package locali. Dove un package è normalmente un'estensione di firefox installata e che dunque rappresenta un insieme di risorse. La sintassi di chrome risponde a *chrome://<package name> /<part>/<file.xul>*. Il foglio di stile deve essere richiamato, solitamente all'inizio del documento xul, in questo modo:

```
<?xml-stylesheet href="findfile.css" type="text/css"?>
```

I fogli di stile sono fondamentali per modificare il rendering degli oggetti grafici, applicando una nuova *skin* alla nostra applicazione. Anche questo è molto semplice, in pratica si tratta di settare dei parametri sui componenti utilizzati:

```
menubar {
  background-color: red;
}
menupopup {
  background-color: green;
}
```

Così facendo il colore di fondo della nostra menubar sarà rossa, mentre la tendina del menupopup verde. A voi il compito di scegliere colori più adatti, ovviamente si possono indicare i colori nel classico formato esadecimale RGB: *#000088* ad esempio.

SCATOLE CINESI?

Nelle interfacce xul la disposizione degli elementi in una window si basa su un layout a box. Utilizzare il *"Model Box"* è piuttosto semplice, basta combinare opportunamente una serie di "scatole" in cui inserire gli elementi, magari distanziandoli fra loro attraverso dei separatori, gli spacers. Esistono due tipi di box: quelli ad orientamento verticale, *<vbox>* e quelli orizzontali, *<hbox>*. Facciamo un rapido esempio:

```
<vbox>
  <button id="yes" label="Yes"/>
  <button id="no" label="No"/>
  <button id="forse" label="Non so"/>
</vbox>
```

In questo modo i pulsanti saranno incolonnati verticalmente, dividendosi tutto lo spazio a disposizione. Ma noi potremmo volere dividere la window in due sezioni, una ad orientamento orizzontale, l'altra verticale, come nell'esempio seguente:

```
<vbox>
  <hbox>
    <label control="login" value="Login:"/>
    <textbox id="login"/>
  </hbox>
  <hbox>
    <label control="pass" value="Password:"/>
    <textbox id="pass"/>
  </hbox>
  <button id="ok" label="OK"/>
  <button id="cancel" label="Annulla"/>
</vbox>
```

Fig. 2: ***Un esempio di utilizzo di VBOX. I tre pulsanti vengono incolonnati verticalmente***

Abbiamo inserito più *<hbox>* all'interno di una grande *<vbox>*. Il risultato è proprio quello che ci aspettavamo, con le *hbox* che si dividono equamente lo spazio con i due bottoni in basso. In HTML lo stesso risultato si ottiene combinando gli elementi *<td>* e *<tr>* in una *<Table>*, ma anche in questo caso il codice è per lo meno poco leggibile.
Ma ancora non abbiamo raggiunto il risultato che cercavamo. Gli elementi nelle *hbox* non sono allineati fra loro. Sarà sufficiente aggiungere due *vbox* opportunamente per migliorare la situazione:

```
<vbox>
  <hbox>
    <vbox>
      <label control="login" value="Login:"/>
```

SUL WEB

XUL, XML User-interface Language, si pronuncia Zool e fa rima con cool. I suoi utilizzi sono molti, anche al di fuori delle applicazioni Mozilla, come possiamo vedere nella pagina principale del progetto:
http://xul.sourceforge.net/

Un Tutorial per l'utilizzo di xul in Firefox, da cui abbiamo tratto alcuni esempi si trova sul sito XulPlanet, portale dedicato a Xul:
http://xul.sourceforge.net/

```
      <label control="pass" value="Password:"/>
    </vbox>
    <vbox>
      <textbox id="login"/>
      <textbox id="pass"/>
    </vbox>
  </hbox>
  <button id="ok" label="OK"/>
  <button id="cancel" label="Annulla"/>
</vbox>
```

Fig. 3a, 3b, 3c, 3d: ***Utilizziamo delle HBOX all'interno di una VBOX. Nella figura 3a i campi non sono opportunamente allineati; nella 3b si. Nella 3c e 3d Vediamo come sia possibile attraverso l'uso di spacer e dell'attributo flex modificare il comportamento della finestra in resize***

Adesso il tutto è molto più ordinato. Notiamo però che se ridimensioniamo la finestra in verticale, gli oggetti rimano tutti della stessa dimensione e tutti allineati verso l'alto. È possibile modificare questo comportamento attraverso l'attributo *flex* che se non indicato, viene impostato a 0. Su tutti i componenti grafici possiamo settare questo attributo e il valore viene interpretato come coefficiente di proprorzione di resize fra gli elementi. Più semplicemente, se abbiamo solo due pulsanti in una *vbox*, se in uno settiamo il *flex* a 2 e nell'altro ad 1, in caso di resize, il primo si ingrandirà il doppio del secondo. L'attributo *flex* è particolarmente utile se utilizzato nell'elemento *spacer*, che altro non è che un bottone invisibile la cui unica utilità è allontanare gli elementi fra loro. Se nell'esempio di prima fra le *hbox* e i bottoni mettiamo uno spacer come questo:

```
<spacer flex="1"/>
```

nell'ingrandire in verticale la finestra, aumenterà solo lo spazio fra i campi di input e i bottoni. Dobbiamo ricordarci di settare, però, il *flex* della *vbox* più esterna a 1, altrimenti la "scatola" non si ridimensionerà.

GLI EVENTI

Un'interfaccia grafica senza alcun controllo è assolutamente inutile. Come abbiamo già visto nell'esempio iniziale, il "motore" di una pagina xul si scrive in Javascript. XUL utilizza il modello degli eventi definito dal consorzio *w3* come modello DOM2. Brevemente, la propagazione dell'evento, un click del mouse per esempio, avviene gerarchicamente, a partire dall'oggetto su cui l'evento è avvenuto, risalendo finché non si trova un elemento in grado di processare l'evento. Vediamo un esempio:

```
<vbox oncommand="loadURL(event);">
<button label="Bottone1" value="<http:
    //www.gmail.com/>" oncommand="load2(event);"/>
<button label="Bottone2" value=
                    "http://www.gmail.com" />
</vbox>
```

Se viene cliccato il primo pulsante, che ha definito un handle per l'evento oncommand viene eseguito il metodo *load2(event)*. Se invece viene cliccato il secondo, che non ha alcun handle per l'evento, questo viene propagato verso l'alto, fino a che non viene trovato un handle adatto. In questo caso quindi verrà processato da *loadURL()* come indicato in *vbox*. Esistono varie tipologie di eventi, vediamo i più utili:

- **onclick:** invocato al click completo di un pulsante del mouse.
- **onmousedown:** invocato quando un un pulsante del mouse viene premuto.
- **onmouseup:** invocato al rilascio del pulsante del mouse.
- **onmouseover:** invocato al passaggio del mouse sull'elemento.
- **onmouseout:** invocato all'uscita del puntatore dall'area di un elemento.
- **oncommand:** invocato quando un menu o un bottone è premuto.
- **onkeypress:** invocato quando un tasto è premuto mentre l'elemento è selezionato

Proviamo ad applicare quello che abbiamo imparato alla nostra barra di navigazione. Abbiamo alcuni menupopup e un button a disposizione. Possiamo provare, per esempio a cambiare la scritta sul bottone, quando il puntatore del mouse gli passa sopra. Al pulsante aggiungeremo un handle per l'evento *onmouseover* e uno per il *mouseout:*

NOTA

La stringa utilizzata per identificare univocamente una estensione è abbastanza lunga da garantirci che non sia mai stata utilizzata, se la scegliamo in maniera del tutto casuale. È presente online una cgi che assegna in questo modo degli id: http://www.hoskinson.net/webservices/guidgeneratorclient.aspx

Jazilla è un clone di Mozilla scritto in Java. Supporta anche lo scripting in xul, grazie al motore JzXUL. http://www.jazilla.org

```
<button id="bottone" label="WebMail" value="http://www
    .gmail.com" onmouseover="cambiascritta(event);"
        onmouseout="ripristinascritta( event);"/>
```

e definiremo due script adatti:

```
<script type="application/x-javascript">
  function cambiascritta(event) {
    var bottone= document.getElementById('bottone');
    bottone.setAttribute('label', "Click Me"); }
  function ripristinascritta(event) {
    var bottone = document.getElementById('bottone');
    bottone.setAttribute('label', "WebMail");
}
```

Gli elementi vengono richiamati attraverso il loro id, che va definito univocamente. Possiamo poi accedere a qualunque attributo dell'oggetto.
Un altro esempio:

```
<menu label="Programmazione" accesskey="p">
    <menupopup>
      <menuitem label="IoProgrammo" accesskey="i"
          value="<http://www.ioprogrammo.it/>" />
    <menuitem label="Javastaff" accesskey="j"
          value="http://www.javastaff.com/" />
    </menupopup>
  </menu>
```

Abbiamo definito per il menu e per tutti i menuitem una *accesskey.* Questo significa che alla pressione del tasto assegnato, viene selezionato l'elemento corrispondente, o si ruota fra gli elementi che hanno la stessa accesskey. In questo modo possiamo creare delle *"shortcut"* per raggiungere velocemente le voci di menu più usate.

LE ESTENSIONI

Fino ad ora abbiamo scritto un unico documento xul, invocandolo direttamente da browser. Questo può essere molto utile, specie se associato ad una generazione dinamica della pagina, per esempio in PHP. Ma chi usa Firefox o Thunderbird ha sicuramente già incontrato le *Extension.* Si tratta di plugins che vanno ad integrarsi nel browser o nel client email, aggiungendo funzionalità altrimenti non presenti. Avrete anche notato che le estensioni sono dei file di estensione *.xpi.* Ci aspetteremo di dover utilizzare strani tool per generare un'estensione.
Invece i file *xpi* altro non sono che file zip rinominati. Per essere precisi in un archivio *xpi* devono essere presenti un file di installazione, *install.rdf* e una cartella chrome che contenga un file *jar.* Nel file *jar* sono presenti tutti i file *xul* e le altre eventuali risorse. Anche questo altro non è che un archivio zip rinominato. Vediamo un esempio di file *install.rdf:*

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02
                              /22-rdf-syntax-ns#"
xmlns:em="http://www.mozilla.org/2004/em-rdf#">
<Description about="urn:mozilla:install-manifest">
  <em:id>{6d863e8e-ec01-4da3-899c-03b994c09b77
                                        }</em:id>
  <em:name>RemoteBookmarks</em:name>
  <em:version>0.1</em:version>
```

L'AUTORE

Luca Mattei è laureando in ingegneria informatica, lavora come progettista software per una Mobile Company che offre la sua consulenza ai gestori di telefonia e alle major del settore ICT. È uno degli amministratori di www.JavaStaff.com, portale dedicato al mondo Java. Potete contattarlo scrivendo a: luca.mattei@javastaff.com

XUL IN SEI PASSI

LA FORMA

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome:
          //global/skin/" type="text/css"?>
<window xmlns="http://www.mozilla.org
    /keymaster/gatekeeper/there.is.only.xul"
        orient="vertical" title="Bookmarks">

application/vnd.mozilla.xul+xml
```

1 **Le prime tre righe del nostro esempio. La prima per evidenziare che il documento che abbiamo davanti è scritto secondo le regole xml. Poi carichiamo il foglio di stile della skin impostata, utilizzando un url del tipo *chrome://*, dal package *global.* Infine apriamo la window, *elemento base* in cui inserire gli altri componenti.**

LA SOSTANZA

```
<menu label="Notizie" accesskey="n">
  <menupopup>
    <menuitem label="Ansa" accesskey=
         "a" value="http://www.ansa.it" />
    <menuitem label="Televideo" accesskey=
      "t" value="http://www.televideo.rai.it
             /nazionale/homenaz.asp" />
  </menupopup></menu>
<button label="WebMail" id="bottone" value=
           "http://www.gmail.com" />
<iframe id="contentFrame" src=
        "http://www.google.it" flex="1" />
```

2 **Alcuni componenti grafici. Abbiamo un menu a tendina con due item, che possiamo a selezionare anche attraverso le shortcut da tastiera indicate nell'attributo accesskey. Seguono un bottone e un frame, in cui abbiamo specificato un id, che ci servirà quando vorremo riferirci ad essi in una funzione di controllo.**

IL CONTROLLO

```
<script type="application/x-javascript">
  function loadURL(event) {
    var contentFrame = document
            .getElementById('contentFrame');
    var url = event.target.getAttribute(
                                  'value');
    if (url) contentFrame.setAttribute(
                             'src', url);}
</script>
...
<menubar id="barra" oncommand=
                      "loadURL(event);">
```

3 **Il motore della nostra applicazione. Senza il controllo, avremmo solo un'interfaccia senza vita. La funzione loadURL() viene invocata quando l'handle impostato nel menubar intercetta l'evento oncommand. Nel javascript otteniamo il Frame attraverso il suo id e andiamo a modificare l'url (src) impostando quello ottenuto dal target dell'evento.**

```
<em:description>Aggiunge link al tuo
              bookmark remoto</em:description>
<em:creator>Luca Mattei</em:creator>
<em:homepageURL>http://www.javastaff.com
                    </em:homepageURL>
<em:iconURL>chrome://RemoteBookmarks
        /RemoteBookmarks.png</em:iconURL>
<em:aboutURL>chrome://RemoteBookmarks
          /content/about.xul</em:aboutURL>
<em:updateURL>http://www.javastaff.com
  /RemoteBookmarks/update.rdf</em:updateURL>
<em:file>
  <Description about="urn:mozilla:extension:file:
                    RemoteBookmarks.jar">
    <em:package>content
              /RemoteBookmarks/</em:package>
    <em:skin>skin/classic
                 /RemoteBookmarks/</em:skin>
  </Description>
</em:file>
<em:targetApplication>
  <Description>
    <em:id>{ec8030f7-c20a-464f-9b0e
                 -13a3a9e97384}</em:id>
    <em:minVersion>0.7</em:minVersion>
    <em:maxVersion>1.5</em:maxVersion>
  </Description>
</em:targetApplication>
</Description>
</RDF>
```

Anche questo è un file XML, come i file *xul*, ma si definisce, invece che un'interfaccia grafica, un manifest per la nostra applicazione. Sono presenti varie informazioni, alcune fondamentali altre meno, tutte abbastanza intuitive. Ci soffermeremo sulle più importanti. Innanzitutto gli id dell'estensione e dell'applicazione target:

```
<em:id>{6d863e8e-ec01-4da3-899c-03b994c09b77
                                  }</em:id>
....
<em:targetApplication>
  <Description>
    <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384
                                  }</em:id>
```

Entrambi gli id sono molto importanti. Il primo identifica univocamente la nostra applicazione, il secondo per quale applicazione l'estensione è stata disegnata. In particolare quella indicata nell'esempio è la stringa che identifica Firefox. Nel file jar dobbiamo includere tutte le risorse, che poi possono essere richiamate attraverso gli url *chrome://*. Il formato di un url di questo tipo è abbastanza semplice:

```
chrome://<package name>/<part>/<file.xul>
```

Il package è il nome del file *jar*, seguito da eventuali directory e dal nome del file scelto.

CONCLUSIONI

Nessuno può dre se Firefox sarà in grado di strappare ad Internet Explorer la palma del browser più diffuso, ma sicuramente *Xul* ha tutte le carte in regola per attuare una vera rivoluzione nel mondo delle interfacce grafiche.

Luca Mattei

LO STILE

```
xul:
<?xml-stylesheet href="style.css" type=
                              "text/css"?>
...
<image id="search"/>
css:
#search {
  list-style-image: url("chrome:
       //findfile/skin/images/search.jpg");}
menubar {background-color: red;}
menupopup {background-color: green;
}
```

4 **I fogli di stile, in Xul come in Html sono fondamentali per modificare il rendering di un documento. Nello xul d'esempio carichiamo un foglio di stile css, dal quale carichiamo il riferimento dell'image, in modo da variare le jpg di una skin, per esempio. Poi variamo l'aspetto di un menu, impostando il colore di sfondo.**

I GIUSTI SPAZI

```
<vbox flex="1">
  <hbox><vbox>
      <label control="login" value="Login:"/>
      <label control="pass" value="Password:"/>
    </vbox><vbox>
      <textbox id="login"/>
      <textbox id="pass"/>
    </vbox></hbox>
<spacer flex="1"/>
  <button id="ok" label="OK"/>
  <button id="cancel" label="Annulla"/>
</vbox>
```

5 **Il Box Model ci permette di ottenere il layout cercato, semplicemente combinando vbox e hbox, l'una dentro l'altra. Ci riserviamo poi degli spazi, e lasciamo che siano questi ad aumentare quando ridimensioniamo la finestra, inserendo uno spacer con l'attributo flex ad 1.**

L'ESTENSIONE

6 **Un file xpi, che può viene riconosciuto da Firefox e Thunderbird come un estensione, è solo un archivio zip compresso. Al suo interno dobbiamo trovare obbligatoriamente il file install.rdf e un file jar, contentente le risorse dell'estensione, compresse in formato zip. Il jar si trova nella cartella chrome.**

Gestire in automatico gli errori 404

Un'applicazione capace di rendere intelligente la gestione degli errori 404, sfruttando ASP, ASP.NET, XML e MySQL, così da aiutare l'utente nella navigazione e lo sviluppatore

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste

basi di ASP, ASP.NET. Precedenti esperienze nell'utilizzo di XML e MySQL

Software

Notepad, MySQL, XMLHTTP per leggere il file XML da ASP, Web service di Google per la correzione grammaticale

Impegno

Tempo di realizzazione

Per tutti i navigatori di Internet, il numero 404 ha ormai assunto il ruolo di errore in assoluto. È il codice di risposta dei server più conosciuto, anche dai meno esperti. 404 identifica un ben preciso errore, il file non trovato. Tanto conosciuto, che nel titolo di questo stesso articolo non abbiamo dovuto neppure esplicitare il suo significato. L'errore 404 è legato a doppio filo con la natura del Web, una Rete che si trasforma, cambia e cresce di minuto in minuto. Le pagine mutano, interi siti vengono trasferiti e i documenti spesso risultano introvabili. Ed è per questo che gli sviluppatori devono porre una grande attenzione per curare questo aspetto.

RENDIAMO COMPRENSIBILE L'ERRORE 404

Il primo passo da fare per gestire al meglio i file non trovati, consiste nel dare più informazioni possibili all'utente che è incappato nell'errore. Distinguiamo due tipi di errore. Il primo generato a causa di una maldestra digitazione dell'URL. Il secondo causato da un link che punta ad una pagina inesistente.

Nel primo caso tenteremo di fornire all'utente un'URL corretto. Nel secondo caso salveremo l'errore in un database per poterlo gestire in maniera programmatica in un secondo momento.

Abbiamo realizzato un programma che ha una struttura logica abbastanza semplice:

- Una volta instanziato dal server l'errore 404, viene richiamata una pagina ASP.
- Questa carica un documento ASP.NET che interroga il Web Service di Google per verificare che l'Url sia stato digitato in maniera grammaticalmente corretta
- In caso negativo (per esempio l'utente ha digitato assistennza invece di assistenza), controlla che l'Url corretto esista e lo propone all'utente.

PERCHÉ NON SALVARE TUTTI GLI ERRORI 404?

All'interno dell'applicazione analizzata nell'articolo, esiste un complesso sistema di verifica degli estremi dell'errore 404 (basato su XML), che se confermati, consentono il salvataggio dell'errore all'interno del database MySQL, a primo acchito potrebbe sembrare un non plus ultra del programma che stiamo scrivendo. A un'attenta analisi però e soprattutto dopo diverse settimane di test, il salvataggio degli errori 404 va centellinato, in particolar modo quando ci troviamo di fronte a un sito di medio o grande traffico. All'interno di siti di queste dimensioni, gli errori 404 si sprecano. Parliamo di decine di file non trovati al giorno e solo una piccola parte possono essere imputati a scorrettezze dello sviluppatore. Spesso sono richieste errate degli spider dei motori di ricerca, oppure di file inesistenti cercati dai software di grabbing, ovvero quei programmi che scaricano un intero sito in pochi minuti. Senza dimenticare che alcuni errori sono causati da malintenzionati che cercano di prendere possesso del server cercando di raggiungere dei file di sistema. Con l'applicazione che abbiamo preparato, siamo in grado di filtrare gli errori 404, così da ottenere solo quelli che ci interessano. Ad esempio, possiamo salvare le pagine non trovate da un particolare spider (così da migliorare il posizionamento del sito), oppure le richieste errate all'interno di una sezione molto importante (come le pagine dedicate allo shopping on line).

- Nel frattempo la pagina ASP controlla un file XML che contiene dei filtri che istruiscono il software sul tipo di errore che deve essere salvato in MySQL.
- Se viene trovata una corrispondenza, l'errore sarà schedulato in un database MySQL.

IL CORE DELL'APPLICAZIONE: IL FILTRO XML

Tutta l'applicazione ruota intorno a un file XML, che stabilisce quali errori salvare e quali no. Nella sua forma più semplice, il documento si presenta così:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<errori>
<errore tipo="404" contiene="false" referrer="true"
        urlcercato="true" useragent="false"
        ip="false">/shop/</errore>
</errori>
```

Vediamo il significato degli attributi, così da poter personalizzare rapidamente l'applicazione.

- **tipo:** contiene il codice 404. Nel caso in cui si volesse estendere la capacità del programma, potremmo inserire qui il codice di altri errori.
- **contiene:** se false, il termine inserito tra i tag <errore> e </errore> dovrà essere presente per salvare il tutto su MySQL. Se true, la sua assenza farà salvare l'errore.
- **referrer:** il termine dovrà essere cercato nell'Url referente, ovvero l'indirizzo che ha portato verso l'errore.
- **urlcercato:** se *true*, si analizzerà l'indirizzo della pagina inesistente ricercata dal client.
- **useragent:** fa riferimento al metodo di identificazione del client (così da sapere se è un browser e quale, un bot, un software di grabbing ecc.)
- **ip:** la corrispondenza verrà cercata sull'IP del visitatore.

MYSQL E XML, I MOTIVI DELLA NOSTRA SCELTA

Il sistema di gestione e monitoraggio degli errori 404, si basa su alcune strutture portanti:

• Una pagina ASP capace di intercettare l'errore.
• Una pagina ASP.NET (VB.NET) che interroga il Web service di Google per la correzione grammaticale
• Un file XML che contiene gli estremi per il salvataggio o meno dell'errore.
• Un database MySQL per l'archiviazione.

L'applicazione è stata scritta appositamente su una macchina Windows, quindi le prime due tecnologie sono quasi vincolate al server. Così come il Web service di Google, un servizio gratuito che consente la correzione grammaticale delle parole. Le ultime due tecnologie scelte però, potevano essere facilmente sostituibili con diverse soluzioni, magari un file di testo e Microsoft SQL Server. Abbiamo però preferito scegliere XML e MySQL per questo tipo di lavoro. XML perché è il formato adatto al contenimento di informazioni per eccellenza, universalmente supportato e quindi facilmente portabile. È facile da capire e rapidamente modificabile, senza necessità di alcun adattamento. Ad esempio, riscrivendo le pagine ASP e ASP .NET in PHP, potremmo trasferire l'applicazione su una macchina Linux senza difficoltà: vedere per credere su http://www.risorse.net/magazine/leggi.asp/id-108 e http://www.risorse.net/magazine/leggi.asp/id-110. La scelta di MySQL, è stata invece supportata dalle caratteristiche che lo hanno reso famoso in tutto il mondo.
È il più veloce a trattare database con una struttura semplice ed è liberamente scaricabile da chiunque. Per chi non lo sapesse poi, MySQL lavora egregiamente anche con le ASP, grazie ai driver MyODBC.

PERCHÉ PERSONALIZZARE LE PAGINE DI ERRORE?

1 Un classico errore 404 senza alcuna informazione di sorta. In questa maniera, l'utente si trova disorientato e la pagina di notifica non da alcun supporto alla navigazione del visitatore.

2 Personalizzando la pagina di errore 404, è possibile far sapere all'utente come comportarsi, dandogli dei suggerimenti o chiedendogli un supporto per risolvere il problema.

3 Con il sistema di gestione degli errori presentato nell'articolo, l'utente ha un supporto unico per proseguire nella navigazione e il Webmaster può risolvere il problema in automatico.

LE API DI GOOGLE

Google rende disponibili, in maniera completamente gratuita fino a 1.000 query al giorno, alcune applicazioni portanti della propria struttura Internet. Tra questi, la possibilità di creare uno strumento di ricerca che sfrutti il motore di Google, oppure un servizio per correggere la grammatica delle parole (quel famoso "Forse cercavi" che appare quando digitiamo male una keyword). Per sfruttare questi Web service, è sufficiente iscriversi attraverso la pagina http://api.google.com/createkey. La registrazione darà diritto, tra gli altri servizi disponibili, alla possibilità di sfruttare una chiave per interrogare il Web service messo a disposizione dal motore di ricerca. Questa chiave, dovrà essere inserita nel file *sistema_404.aspx* dell'applicazione analizzata nell'articolo. Google riesce a correggere anche le frasi che hanno una struttura sintattica non corretta. Se inoltriamo una ricerca digitando come keywords: */cavallllo /addestramento* (si notino le tre elle), Google ci risponderà con *"Forse cercavi: /cavallo/addestramento"*. Sfruttando questa capacità, possiamo inoltrare l'URL al nostro Web service senza per questo dover scomporre l'indirizzo che ha generato l'errore.

Ammettiamo di voler salvare tutti gli errori 404 che provengono dai principali motori di ricerca (Google, Virgilio, Msn). Potremo personalizzare il file XML come segue:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<errori>
<errore tipo="404" contiene="false" referrer="true"
        urlcercato="false" useragent="false" ip="false"
                                   >google</errore>
<errore tipo="404" contiene="false" referrer="true"
        urlcercato="false" useragent="false"
                          ip="false">msn</errore>
<errore tipo="404" contiene="false" referrer="true"
        urlcercato="false" useragent="false" ip="false"
                                  >virgilio</errore>
</errori>
```

Così scritto, il file XML indicherà all'applicazione di salvare tutti gli errori 404 generati dai referrer che contengano nell'indirizzo i termini google, oppure msn o ancora virgilio.

IL FILE ASP.NET E IL WEB SERVICE DI GOOGLE

La pagina che corregge la grammatica dell'Url non trovato, scritta in ASP.NET, si basa come abbiamo già visto, su un Web service fornito gratuitamente da Google a seguito di una semplice iscrizione.

Il documento, realizzato in VB.NET, propone un indirizzo grammaticalmente corretto, solo quando questo è presente sul server, sfruttando i metodi *System.Io.Directory.Exists()* o *System.Io.File .Exists()*.

Ecco la sua struttura sintattica:

```
<%@ Assembly Src="search.vb" %>
<script runat="server">
sub Page_Load(sender as object, e as EventArgs)
Response.ContentType="text/plain"
Response.addHeader("pragma", "nocache")
```

COME IMPOSTARE LA PAGINA D'ERRORE IN IIS

1 **Il primo passo Ë modificare la pagina di risposta degli errori 404.** Dal Pannello di Controllo > Strumenti di Amministrazione > Gestione Servizi Internet Microsoft. Dal proprio sito Web, tasto destro del mouse: Propriet‡ > Errori personalizzati. Posizionarsi sul 404 e fare click su Modifica propriet‡. Scegli URL e specifica l'indirizzo della nuova pagina di notifica.

2 **La nuova pagina di risposta per gli errori 404, analizza un file XML che** contiene tutti i parametri per segnalare all'applicazione se salvare o meno l'errore generato dall'utente. Il file XML Ë in grado di instradare la segnalazione del problema in base all'URL cercato, l'URL referente, l'User Agent utilizzato dal navigatore (o da software automatici) e l'IP del client.

3 **In base ai parametri inseriti nel file XML, l'applicazione salva gli** estremi dell'errore 404 generato dal client, all'interno di un database MySQL, la cui struttura Ë composta da 9 campi: id, codice_err (es. 404), referrer (1 o 0, sÏ o no), url_ref (URL referente), url_err (URL cercato), ip, data_ora, user_agent, QuanteVolteVisto (numero di errori uguali).

USARE UN WEB SERVICE DI GOOGLE

1 **Spesso un errore 404 viene generato quando Ë l'utente stesso a digitare male l'indirizzo che sta cercando (ad esempio omettendo qualche carattere dell'URL).**

```
<%@ Assembly Src="search.vb" %>

<script runat="server">
sub Page_Load(sender as object, e as EventArgs)

Response.ContentType="text/plain"
Response.addHeader("pragma", "nocache")
Response.Expires=-2

Dim indirizzo As String
Dim dominio As String
dominio = "http://" & Request.ServerVariables("SERVER_NAME")
indirizzo = Request.Querystring(0)
indirizzo = Replace(indirizzo,dominio,"")

Dim s As New GoogleSearchService()
Dim suggestion As String = s.doSpellingSuggestion("iYZKPLhQFHJ0dVKBH
If Len(suggestion) > 0 Then
      If System.Io.Directory.Exists(Server.MapPath(suggestion)) OrEls
            Response.write(String.Format(VbCrLF & "document.write(""I
dominio & "{0}'>" & dominio & "{0}</a>"");", suggestion))
      Else
            Response.write("document.write(""Non esiste una pagina si
disponibile."")")
```

2 **In caso di indirizzi digitati male, l'applicazione si appoggia ad ASP.NET e alle API di Google per correggere l'URL e proporne uno grammaticalmente a posto.**

```
If Len(suggestion) > 0 Then
  If System.Io.Directory.Exists(
      Server.MapPath(suggestion)) OrElse
    System.Io.File.Exists(Server.MapPath(
                        suggestion)) Then
Response.write(String.Format(VbCrLF &
                "document.write( ""Il sistema
  automatico ti consiglia di provare: <a
href='" & dominio & "{0}'>" & dominio &
              "{0}</a>"");", suggestion))
```

3 **Il programma verifica che l'URL grammaticamente corretto in automatico dal Web Service di Google esista nel server. In caso positivo, lo consiglia al visitatore.**

```
Response.Expires=-2
Dim indirizzo As String
Dim dominio As String
dominio = "http://" & Request.ServerVariables(
                                "SERVER_NAME")
indirizzo = Request.Querystring(0)
indirizzo = Replace(indirizzo,dominio,"")
Dim s As New GoogleSearchService()
Dim suggestion As String = s.doSpellingSuggestion(
    "qui_la_chiave_della_api_google", indirizzo)
If Len(suggestion) > 0 Then
  If System.Io.Directory.Exists(Server.MapPath(
        suggestion)) OrElse System.Io.File.Exists(
                   Server.MapPath(suggestion)) Then
    Response.write(String.Format(VbCrLF &
           "document.write(""Il sistema automatico
      ti consiglia di provare: <a href='" & dominio &
           "{0}'>" & dominio & "{0}</a>"");",
                                      suggestion))
  Else
    Response.write("document.write(""Non esiste
             una pagina simile a quella ricercata.
           Nessun suggerimento disponibile."")")
      End If
Else
  Response.write("document.write(""Il sistema
  automatico non rintraccia suggerimenti. Il problema
                    verra' gestito dai tecnici"");")
End If

End Sub
</script>
```

In questa maniera, solo se il file o la directory corretta dal Web service di Google esistesse, verrebbe proposta all'utente come valida alternativa. Evitando quindi inutili correzioni.

Roberto Abbate

FARE UN FILTRO CON XML

```
- <!--
  Salvare solo gli errori 404 generati da GoogleBot
  -->
- <errori>
    <errore tipo="404" contiene="false" referrer="false" urlcercato="false" useragent="true" ip="false">google</errore>
  </errori>
```

1 **La grande flessibilità dell'applicazione ci consente di verificare gli errori 404 che vogliamo. Ad esempio, possiamo personalizzare il file XML in maniera tale da archiviare solo gli errori generati da GoogleBot, lo spider di Google. In questa maniera, risolvendo i problemi che il bot riscontra, potremo migliorare il ranking, ovvero il posizionamento, del nostro sito.**

```
- <!--
  Salvare gli errori 404 generati nella cartella /shop/
  -->
- <errori>
    <errore tipo="404" contiene="false" referrer="true" urlcercato="true" useragent="false" ip="false">/shop/</errore>
  </errori>
```

2 **Si sa che all'interno di un sito, esistono sezioni più importanti di altre. Per chi possiede un'attività di shopping on line, potrebbe voler essere assolutamente sicuro che quelle pagine non contengano errori 404. Per monitorarle, può personalizzare l'applicazione, modificando il file XML in maniera tale da intercettare le pagine non trovate che contengano la cartella shop nell'Url.**

```
- <!--
  Salvare tutti gli errori 404 tranne quelli generati dall'IP 80.128.1.42
  -->
- <errori>
    <errore tipo="404" contiene="true" referrer="false" urlcercato="false" useragent="false" ip="true">80.128.1.42</errore>
  </errori>
```

3 **Grazie ai differenti attributi presenti nel file XML, è possibile monitorare qualunque errore 404. Volendo, potremmo allargare il campo di ricerca, escludendo un tipo di errore ma salvando tutti gli altri. Per esempio, è possibile salvare tutti gli errori 404 tranne quelli causati da un particolare IP (che potrebbe essere quello del webmaster che sta facendo i test).**

Disegnare con PHP

Punti, linee, immagini, forme geometriche, spesso e volentieri arricchiscono le nostre pagine. Come si può utilizzare PHP per creare contenuti grafici? Ve lo spieghiamo noi

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Principi di PHP

Software
PHP

Impegno

Tempo di realizzazione

D'accordo, non c'è certo bisogno di una lunga introduzione per spiegare cosa vogliamo ottenere con questo articolo. Perciò è utile iniziare subito con un esempio pratico: disegnare un punto sullo schermo. Considerate il seguente spezzone di codice:

```
<?
  Header("Content-Type: image/png");
  $immagine = ImageCreate(300,300);
  $nero = imagecolorallocate($immagine,0,0,0);
  $bianco=imagecolorallocate($immagine,255,255,0);
  imagesetpixel($immagine,150,150,$bianco);
  imagepng($immagine);
  imagedestroy($immagine);
?>
```

Queste poche righe di codice disegnano sullo schermo un quadrato nero con un puntino bianco in mezzo. Avete già i brividi no? Cerchiamo di capirci qualcosa. La prima riga informa il browser che quello che seguirà è un'immagine di tipo PNG. Se omettessimo questa riga, vedremmo a schermo il contenuto binario di quanto creato dalle funzioni che seguono e non l'immagine vera e propria, allo stesso modo se al posto del contenuto informativo *"PNG"* utilizzassimo *"JPG"* l'immagine non verrebbe costruita correttamente, in quanto l'istruzione *imagepng(...)* produce un'immagine di tipo *PNG* e non *JPG*. Per cui il primo passo per far si che l'immagine sia visualizzata correttamente è dichiarare il suo header corretto. La seconda linea definisce una "tela" di dimensioni 300x300 pixel e identificata dalla variabile *$immagine*. Al momento della sua definizione la tela è completamente vuota. Le linee contenenti la funzione *imagecolorallocate(...)* definiscono i colori da usare sulla tela di nome *"Immagine"*. I colori vengono espressi nella loro combinazione RGB, ovvero red, green, blue. Rosso, verde e blue sono i colori base, combinando i quali è possibile ottenere tutti gli altri. Il primo colore ad essere definito per la tela ne diventa anche il colore di sfondo. Nel nostro caso sarà il nero. La funzione *imagesetpixel(...)* disegna un punto sullo tela *"immagine"*, lo posiziona alle coordinate *x* e *y* 150,150 e lo colora con il colore che abbiamo denominato *"$bianco"*. La funzione *imagepng(...)* disegna fisicamente l'immagine sulla tela, e infine la funzione *imagedestroy* libera la memoria che fino ad ora avevamo impiegato per realizzare i nostri scopi. Il risultato non è eclatante, ma sono già presenti gli elementi base per usare correttamente le estensioni grafiche di PHP.

COME INIZIARE

Le funzioni per la gestione delle immagini sono inserite nel contesto delle librerie GD. Per utilizzarle dovete avere abilitato queste librerie. In ambiente windows potete semplicemente decommentare la linea *extension=php_gd2.dll*.
In ambiente linux dovete avere compilato il supporto alle GD all'interno del PHP.

DISEGNARE UNA FUNZIONE

L'idea è molto semplice, visto che abbiamo disegnato un punto sullo schermo, disegniamo una serie di punti che rappresentano una funzione matematica. L'esempio è il seguente:

```
Header("Content-Type: image/jpeg");
$immagine = ImageCreate(300,300);
$nero = imagecolorallocate($immagine,0,0,0);
$bianco=imagecolorallocate($immagine,255,255,0);
for ($x=1;$x<100;$x++) {
  $y=pow($x,2);
  $yt=300-$y;
  imagesetpixel($immagine,$x,$yt,$bianco);}
imagejpeg($immagine);
```

```
imagedestroy($immagine);
```

Sicuramente avrete notato che abbiamo cambiato qualche piccolo particolare. Prima di tutto l'header e la relativa funzione di disegno non sono più associati al formato *PNG* ma al formato *JPG*. Poi abbiamo aggiunto un ciclo di *for* che disegna sulla tela un punto aggiungendolo ai precedenti e ricavandolo da una funzione matematica. Ora se proverete ad eseguire questo spezzone di codice vi accorgerete che non è per niente efficiente perchè disegna la funzione in modo discontinuo, stiamo utilizzando infatti degli interi e non dei numeri reali, perciò ovviamente vedremo a schermo solo i punti notevoli, ma in questo articolo non siamo molto preoccupati della precisione matematica, quello che ci importa è introdurvi al mondo delle immagini con PHP, ed in questo esempio la cosa più interessante da notare è che abbiamo effettuato una traslazione dell'origine degli assi. Chi mastica un po di matematica sa che l'origine degli assi, ovvero lo 0,0 è sempre fissato in basso a sinistra, in questo caso invece lo 0,0 è collocato nell'angolo sinistro superiore, perciò per rappresentare la nostra immagine in modo corretto, abbiamo traslato gli assi.

IL NOSTRO PRIMO EFFETTO

Nel prossimo esempio sfrutteremo le conoscenze acquisite per leggere un'immagine preesistente, e rimpiazzare ogni punto dell'immagine con un punto colorato in modo differente, gli effetti realizzabili con questa tecnica sono piuttosto interessanti.

```
<?
Header("Content-Type: image/jpeg");
$imagefile="cat.jpg";
$image=imagecreatefromjpeg($imagefile);
$imagesize=getimagesize($imagefile);
$w=$imagesize[0];
  $h=$imagesize[1];
  for ($x=0; $x<$w; $x++) {
      for ($y=0; $y<$h; $y++) {
          $colore=imagecolorat($image, $x, $y);
           if ($x < $y) {
              $rgb=imagecolorsforindex($image, $colore);
              if ($rgb['red']+10 < 255 )
                              $rgb['red']=$rgb['red']-60;
              if ($rgb['green']+60 < 255 )
                       $rgb['green']=$rgb['green']+60;
              if ($r gb['blue']-30 < 255 )
                             $rgb['blue']=$rgb['blue']-30;
               $colore=imagecolorallocate($image,
           $rgb['red'],$rgb['green'],$rgb['blue']);}
           imagesetpixel($image,$x,$y,$colore);}}
  imagejpeg($image);
  imagedestroy($image);
?>
```

In questo esempio abbiamo sostituito il colore di tutti i pixel sotto la bisettrice dell'immagine con un colore settato manualmente. L'effetto è quello in **Figura 1**. Chiaramente anche in questo caso non pretendiamo di avere creato un effetto sorprendente, ma se avete un minimo di esperienza con le formule di sostituzione del colore, potrete senza dubbio realizzare effetti molto complessi. Le funzioni introdotte con questo spezzone di codice non sono poche. Si parte con *imagecreatefromjpeg(...)* questa funzione crea un'immagine in memoria prendendola direttamente da un file. È molto differente dalla funzione *ImageCreate(...)*, che abbiamo visto in precedenza. In questo caso non prendiamo un'immagine vuota per poi riempirla. Viceversa prendiamo un'immagine da un file e la copiamo direttamente in memoria, associandola ad una variabile. Come sempre esistono anche le funzioni *imagecreatefromjpng()*, etc. Tutto dipende dal formato del file che state utilizzando. La seconda funzione interessante è *getimagesize(...)*; che prende in input un'immagine e restituisce un array contenente la larghezza e l'altezza dell'immagine. La funzione *imagecolorat(...)* restituisce il colore del pixel ricevuto in input, infine *imagecolorsforindex(...)* restituisce un array contenente il valore *Red, Green, Blue* del singolo colore RGB. Il funzionamento dell'esempio di cui sopra è veramente banale. Viene creata un'immagine partendo da un file. Viene eseguito un ciclo di *for.*
Per ciascun ciclo viene recuperato il colore del pixel corrente, se il pixel si trova al di sotto della bisettrice dell'immagine viene ridisegnato allocando un colore diverso, altrimenti viene disegnato come sempre. Si tratta di un esempio banale, ma giocando con la sola allocazione dei colori è facile ottenere effetti migliori.

DISEGNARE CIRCONFERENZE E ARCHI

Chiaramente una linea o una forma geometrica può essere rappresentata disegnando i singoli punti che la compongono, oppure esplicitando la funzione matematica che la descrive, tuttavia questo non è certo un buon metodo. Esistono funzioni PHP predefinite che consentono di disegnare le forme geometriche. *Disegnare una linea* ad esempio corrisponde a:

```
Header("Content-Type: image/jpeg");
$immagine = ImageCreate(300,300);
$nero = imagecolorallocate($immagine,0,0,0);
$bianco=imagecolorallocate($immagine,255,255,0);
ImageLine($immagine,0,0,300,300,$bianco);
```

SALVARE LE IMMAGINI

Tutte le immagini generate possono essere salvate su Hard Disk. Si può effettuare questa operazione specificando il nome dell'immagine come argomento della funzione che la rappresenta. Ad esempio: ***imagejpeg($immagine, "miaimmagine.jpg");*** **Si può anche specificare un terzo argomento opzionale. Un intero compreso fra -1 e 100 che rappresenta la qualità dell'immagine.**

***Fig. 1:** Il gatto prima e dopo la cura, notate nella seconda immagine l'applicazione dell'effetto*

```
imagejpeg($immagine);
imagedestroy($image);
```

Allo stesso modo *creare un cerchio* è sufficientemente semplice

```
Header("Content-Type: image/jpeg");
$immagine = ImageCreate(300,300);
$nero = imagecolorallocate($immagine,0,0,0);
$bianco=imagecolorallocate($immagine,255,255,0);
ImageArc($immagine,150,150,60,60,0,360,$bianco);
imagejpeg($immagine);
imagedestroy($immagine);
```

Si noti però che la funzione che effettua il disegno ha un nome particolare, ovvero *ImageArc(...)*, il che lascia presupporre che può essere utilizzata per scopi diversi dalla rappresentazione di una semplice circonferenza. Infatti la dichiarazione completa di *ImageArc(...)* è la seguente:

```
ImageArc(im, x,y,larghezza,altezza,inizio, fine, colore)
```

Dove *im* è il puntatore all'immagine. *x* e *y* sono le coordinate del punto centrale dell'arco, larghezza e altezza rappresentano la larghezza e l'altezza della forma geometrica da disegnare. Non corrispondono al raggio, ma sono due misure che ci consentono di creare anche ellissi o forme non perfette. Inizio e Fine rappresentano i gradi di inizio e fine dell'arco. Ad esempio volendo disegnare un quarto d'arco si potrebbe usare la seguente:

```
ImageArc($immagine,150,150,60,120,0,90,$bianco);
```

si tratterebbe di un arco non perfetto, in quanto larghezza e altezza differiscono.

DISEGNARE RETTANGOLI E POLIGONI

La funzione che assolve al compito di disegnare un rettangolo è la *ImageRectangle*. La sua definizione completa è:

```
ImageRectangle(immagine, x,y,x1,y1,colore)
```

Dove immagine è come sempre un puntatore alla tela che abbiamo creato, *x* e *y* rappresentano l'angolo superiore sinistro del rettangolo e *x1,y1*, l'angolo inferiore destro. Colore, come sempre è un colore che abbiamo allocato in precedenza e che utilizzeremo per disegnare le linee che formano il rettangolo. Ad esempio:

```
Header("Content-Type: image/jpeg");
$immagine = ImageCreate(300,300);
$nero = imagecolorallocate($immagine,0,0,0);
$bianco=imagecolorallocate($immagine,255,255,0);
ImageRectangle($immagine,10,10,290,290,$bianco);
imagejpeg($immagine);
imagedestroy($immagine);
```

disegna una cornice bianca all'interno della nostra tela.

DISEGNARE IL TESTO

La funzione che si occupa di disegnare il testo è la *ImageString*. La sua dichiarazione completa è la seguente:

```
ImageString(immagine,font,x,y,testo, colore)
```

Fig. 2: *un esempio di come il testo viene posizionato sull'immagine*

Dove *immagine* è il solito puntatore, *font* è il font con cui vogliamo disegnare il testo, *testo* è il testo da scrivere e *colore* è il colore che abbiamo allocato per la stringa. Ad esempio volendo sovrascrivere con un testo una immagine prima di proiettarla in output, in modo molto meno raffinato rispetto a quanto illustrato nell'articolo sul *WaterMarking* presente in questo stesso numero di ioProgrammo, si potrebbe utilizzare la seguente tecnica:

```
Header("Content-Type: image/jpeg");
$imagefile="./einstein2.jpg";
$immagine=imagecreatefromjpeg($imagefile);;
$nero = imagecolorallocate($immagine,0,0,0);
$bianco=imagecolorallocate($immagine,255,255,0);
ImageString($immagine,10,10,5,"Immagine di
                                    Einstein",$bianco);
ImageString($immagine,10,10,20,"in bianco e
                                    nero",$bianco);
imagejpeg($immagine);
imagedestroy($immagine
```

È importante notare che il *font* viene espresso con un un intero da uno a cinque che rappresenta uno dei font embedded nel sistema. Una funzione un po più precisa, che consente un posizionamento più accurato del testo e anche di utilizzare un font true type è la *imagettftext(...)*, che corrisponde alla seguente sintassi

```
imagegettftext(immagine, size, angolo, x, y, colore,
```

NOTA

POSSIBILI ERRORI

Se provate a generare un'immagine al volo e a miscelarla in altri output ad esempio delle stringhe, otterrete degli errori di "Cannot Modify Header Information", questo perché con la funzione header avete dichiarato che quello che segue è un'immagine, non potete miscelarlo a niente altro. Potete usare un trucco del genere per aggirare l'ostacolo

```
imagepng($immagine,
         "./pippo.png");
imagedestroy($immagine);
echo "Questa è
  l'immagine pippo<br>";
echo '<img src=
        "pippo.png">';
```

oppure richiamare il file php da un file html come segue:

```
Questa è l'immagine
        "Pippo"<br>
<img src="./index.php>
```

in questo caso in *index.php* dovete mantenere l'header

```
fontfile, testo)
```

Fig. 3: **La funzione imagegettftext ci consente un controllo maggiore su font e posizionamento delle stringhe**

Dove *size* è un intero contenente le dimensioni da attribuire al font per disegnare il testo, angolo è l'angolo con cui la stringa deve essere proiettata a schermo, espresso in gradi a partire da 0. *Fontfile* è il file .ttf che contiene, il font. Il resto è facilmente intuibile. Ad esempio:

```
Header("Content-Type: image/jpeg");
$imagefile="./einstein2.jpg";
$immagine=imagecreatefromjpeg($imagefile);;
$nero = imagecolorallocate($immagine,0,0,0);
$bianco=imagecolorallocate($immagine,255,255,255);
imagettftext($immagine, 15,-15,10,20,$bianco,
        "FreeSerif.ttf","Einstein in bianco e nero");
imagejpeg($immagine);
imagedestroy($immagine);
```

stampa un'immagine di einstein con un'etichetta obliqua di colore bianco in font FreeSerif.

COPIARE E RIDIMENSIONARE LE IMMAGINI

Tra le funzioni più interessanti, fra quelle esposte da PHP, per la manipolazione di immagini ci sono quelle relative a copia e ridimensionamento. Sono molto utili ad esempio quando si vuole eseguire un upload di un'immagine da remoto verso un server a dimensione fissa, o quando si vuole creare un thumbnail partendo da un'immagine più grande, e perché no quando si vuole costruire un'immagine composita. Ad esempio potremmo volere attribuire un voto ad articolo rappresentandolo con delle stelline, l'immagine contenente il numero di stelle corretto potrebbe essere costruita dinamicamente. La funzione che copia un immagine in un'altra è la imagecopy(...) , risponde alla seguente sintassi:

```
imagecopy(dst,src,dst_x,dst_y,src_x,src_y,src_w, src_h)
```

dove *dst*, e *src* rappresentano rispettivamente l'immagine destinazione e l'immagine sorgente.
Al solito è utile procedere con un esempio:

```
Header("Content-Type: image/jpeg");
$filesrc="./einstein2.jpg";
$src=imagecreatefromjpeg($filesrc);
$imagesize=getimagesize($filesrc);
$dst=imagecreate(300,300);
imagecopy($dst,$src,100,100,70,90,100,80);
imagejpeg($dst);
imagedestroy($dst);
imagedestroy($src);
```

questo spezzone di codice ritaglia gli occhi di einstein e li disegna al centro di un'immagine vuota sullo schermo. È interessante, ma lo è ancor di più la funzione *imagecopyresized(...)* , che ci consente di scalare l'immagine, utilissima ad esempio per effettuare uno zoom. La funzione *imagecopyresized(...)*, risponde alla seguente sintassi:

```
imagecopyresized(dst,src,dst_x,dst_y,src_x,src_y,
                        dst_w,dst_h,src_w,src_h)
```

Volendo zoomare sugli occhi di einstein si potrebbe utilizzare il seguente esempio:

```
Header("Content-Type: image/jpeg");
$filesrc="./einstein2.jpg";
$src=imagecreatefromjpeg($filesrc);
$imagesize=getimagesize($filesrc);
$dst=imagecreate(300,300);
imagecopyresized($dst,$src,0,0,70,80,300,300,100,100);
imagejpeg($dst);
imagedestroy($dst);
imagedestroy($src);
```

Il che è utile, ma produce un'immagine un po sgranata, frutto dello zoom, un ulteriore miglioria si può ottenere con la funzione *imagecopyresampled(...)* che ha esattamente la stesso scopo della precedente, ma in più aggiunge il fatto che i pixel vengono interpolati e l'immagine ha un risultato migliore, il solito zoom sugli occhi di Einstein viene prodotto dal seguente codice di esempio:

```
Header("Content-Type: image/jpeg");
$filesrc="./einstein2.jpg";
$src=imagecreatefromjpeg($filesrc);
$imagesize=getimagesize($filesrc);
$dst=imagecreatetruecolor(300,300);
imagecopyresampled($dst,$src,0,0,70,80,300,300,
                                        100,100);
imagejpeg($dst);
imagedestroy($dst);
imagedestroy($src);
```

da notare che questa volta abbiamo utilizzato anche la funzione *imagecreatetruecolor(...)*, il che crea un'immagine di destinazione capace di allocare più di 255 colori ed evita che possano generarsi problemi nel caso in cui l'operazione richieda una palette più ricca del normale.

DIFFERENZE FRA JPEG, PNG, GIF

Il formato GIF è un formato senza perdita di qualità a bassa compressione, il metodo di compressione è lo LZW. Supporta fino a 256 colori, la trasparenza delle immagini e le animazioni. Ha il grande svantaggio di essere un formato proprietario per cui non è sempre possibile utilizzarlo, in ogni caso le GD non lo supportano in modo adeguato.

Il formato JPEG è un formato con alta compressione supporta 16.777 .216 colori, non supporta trasparenze e animazioni, tuttavia è largamente utilizzato su internet, poiché le dimensioni dei file generati con questo formato sono veramente piccole e questo le rende particolarmente adatte al web anche in caso di banda ridotta

Il formato PNG è un formato con bassa compressione e alta qualità, supporta 16.777.216 colori, utilizza il metodo L777 e sta lentamente prendendo il posto del formato GIF. In ambito OpenSource è particolarmente utilizzato.

Questi pazzi pazzi thread

Usare i thread di Java può sembrare abbastanza semplice Ma ci sono alcune cose importanti da sapere se non si vogliono commettere errori clamorosi. Questo mese ne vedremo alcune

Utilizza questo spazio per le tue annotazioni

Immaginiamo di dover scrivere un videogioco. Ovviamente dovremo ottimizzarlo usando la programmazione concorrente. Uno degli oggetti del gioco è un satellite:

```
import java.awt.Point;
public class Satellite implements Runnable {
    private final static int MAX_POS = 10000;
    private Point _posizione = new Point(0, 0);
    private void muovi() {
        verificaCollisioni();
        _posizione.x++;
        _posizione.y++; }
    public Point getPosizione() {
        return (Point)_posizione.clone();}
    public void run() {
        long tempoIniziale = System.currentTimeMillis();
        for(int i = 0; i < MAX_POS; i++)
            muovi();
        long tempoFinale = System.currentTimeMillis();
        System.out.println("Tempo totale: " +
       ((double)tempoFinale - tempoIniziale) / 1000);}
    private void verificaCollisioni() {
        aspetta();}
    private static void aspetta() {
        try {Thread.sleep(5);
        } catch (InterruptedException e) {
            e.printStackTrace();}}}
```

Cerchiamo di capire pezzo per pezzo come funziona questa classe. Il metodo *verificaCollisioni()* è solo uno *"stub"*, cioé un segnaposto per il codice che dobbiamo ancora scrivere. In un ipotetico futuro controllerà se il Satellite è andato a sbattere contro qualcosa e produrrà colorate esplosioni. Per ora contiene solo una chiamata a *Thread.sleep()*, che sospende il thread attuale per un certo tempo – in questo caso per cinque millisecondi. *Thread.sleep()* può anche lanciare una *InterruptedException*, che nel nostro caso ignoriamo beatamente. Ho dichiarato static il metodo *aspetta()*, perché ho appena visto nella mia palla di vetro che tra poco lo vorrò chiamare anche dal *main()*. L'unico campo del Satellite è la posizione, conservata in un *java.awt.Point*. La classe *Point* è una semplice struttura che contiene due interi: la *x* e la *y* su un piano cartesiano. I client possono leggere la posizione del Satellite con *getPosizione()*, che non restituisce la posizione originale, ma un suo clone. In questo modo impediamo che un client modifichi la posizione originale. Di solito questo si chiama "incapsulamento"; ma in questo caso, trattandosi di un semplice esempio, potrebbe anche chiamarsi "piccola concessione alla mia naturale paranoia". Il Satellite è un Runnable, quindi lo si può usare per costruire un thread. Questa è una buona cosa: se tutti gli oggetti del nostro gioco si muovono in thread diversi, potranno muoversi tutti contemporaneamente (in realtà di solito i giochi non funzionano così, ma questo modo di procedere è ideale per il nostro esempio). Il metodo *run()* contiene un

COMPILARE E LANCIARE GLI ESEMPI CON ECLIPSE

1 **Crea una cartella per il progetto. In questa cartella crea una cartella di nome *src* e scompattaci dentro il file ZIP che contiene gli esempi di questo mese.**

2 **Da Eclipse, scegli *File->New->Project*. Seleziona *Java Project* e premi *Next*.**

3 **Dai un nome al progetto. Seleziona *Create project at external location,* e come Directory seleziona la directory che hai creato al punto 1. Seleziona *Use project folder as root for source and class files.* Premi *Finish*.**

4 **Nel Package *Explorer* di Eclipse, dovresti vedere il nuovo progetto. Per lanciare una classe, selezionala (a patto che abbia un *main()*) e dal menu contestuale scegli *Run->Java Application.***

ciclo che chiama il metodo *muovi()*, che a sua volta incrementa le coordinate e *y* finché il satellite non raggiunge la posizione (10000, 10000). Chi ha studiato geometria può dire che il Satellite parte dall'origine degli assi e si muove lungo la bisettrice del primo quadrante; chi non l'ha studiata può dire, più semplicemente, che il Satellite si muove in diagonale verso un punto in alto a destra. Il metodo *muovi()* controlla eventuali collisioni ogni volta che sposta il Satellite. Alla fine, *run()* stampa il tempo richiesto al Satellite per raggiungere la sua meta.
Ora scriviamo un *main()* che attiva il Satellite. Mi piace l'idea di una torre di controllo per satelliti, quindi questo *main()* verifica anche che il movimento del Satellite sia quello atteso:

```
public static void main(String[] args) {
    Satellite sat = new Satellite();
    Thread threadSat = new Thread(sat);
    threadSat.setDaemon(true);
    threadSat.start();
    int numeroControlli = 0;
    Point pos = sat.getPosizione();
    while(pos.x < MAX_POS) {
        if(pos.x != pos.y)
          throw new RuntimeException("Deviazione dalla
            rotta! Pos.: (" + pos.x + ", " + pos.y + ")");
        aspetta();
        pos = sat.getPosizione();
        numeroControlli++; }
    System.out.println("Numero controlli: " + pos);}
```

MAGIA DEI THREAD

Questo programma costruisce un Satellite, lo lancia in un thread separato e ne verifica continuamente la posizione fin quando il Satellite non ha raggiunto la meta. Visto che il Satellite si muove lungo la bisettrice, ci aspettiamo che la sua posizione *x* sia sempre uguale alla sua posizione *y*. Questo è il nostro controllo di integrità sull'oggetto. Se questo controllo fallisce, il *main()* lancia un'eccezione. Un'altra riga interessante è quella che chiama *Thread.setDaemon()*. Normalmente, un programma Java non termina fino a quando tutti i thread non sono terminati. I cosiddetti daemon sono thread "di servizio", che cioé non impediscono al programma di terminare. Quindi la regola è che il programma termina quando sono terminati tutti i thread che non sono daemon. Il thread che muove il Satellite è un daemon, perché se il *main()* termina con un'eccezione vogliamo che il programma si interrompa subito. In caso contrario, la JVM non terminerebbe finché il Satellite non è giunto a destinazione. Ecco cosa è successo la prima volta che ho lanciato il programma:

```
java.lang.RuntimeException: Deviazione dalla rotta!
                                   Pos.: (179, 178)
  at it.ioprogrammo.threads2.esempio1.Satellite.main(
                                   Satellite.java:40)
```

Quindi esistono dei casi nei quali *x* e *y* non sono uguali come ci aspetteremmo. A volte questo problema si verifica immediatamente, a volte più tardi. In qualche caso, il Satellite riesce a raggiungere la sua destinazione senza che il *main()* trovi nessuna incongruenza. Questa imprevedibilità non fa che rendere il problema peggiore. In un programma "vero", i dati inconsistenti possono causare degli autentici disastri. Un vero satellite sarebbe già uscito dall'orbita e precipitato su un autobus pieno di programmatori in vacanza. Il bug che abbiamo visto dipende dal fatto che la scrittura e la lettura della posizione del Satellite avvengono in due thread diversi, e non sono atomiche. È possibile che *getPosizione()* legga le coordinate proprio mentre il metodo *muovi()* la sta cambiando. Se la coordinata *x* è già cambiata, ma la coordinata *y* deve ancora cambiare, il risultato è che leggiamo (o in questo caso cloniamo) l'oggetto *Point* mentre è in uno stato inconsistente, con *x* e *y* che hanno temporaneamente valori diversi e quindi illegali. Questo rischio è sempre presente quando più thread accedono liberamente allo stesso dato. Questo non vale solo per gli oggetti, ma anche per le variabili primitive: quando ci sono i thread di mezzo, i dati che vengono letti o scritti possono assumere valori imprevisti. Ad esempio, un thread può modificare il valore di una variabile booleana senza che un secondo thread si accorga del fatto che il valore è cambiato. Per spiegare come queste cose possano essere possibili dovremmo entrare nei dettagli del funzionamento di Java, cosa che non abbiamo lo spazio per fare. Ci limitiamo a

IL RITORNO DELLE LUMACHE DA CORSA

Questo box è dedicato a chi ha letto l'articolo del mese scorso, che terminava con un problema. Avevamo scritto un simulatore di corsa di lumache. Il problema era che una Lumaca può annunciare la propria vittoria per poi veder assegnare il titolo ad una concorrente. Il bug deriva dal fatto che ciascuna Lumaca, una volta arrivata al traguardo, fa due cose: stampa la propria vittoria sullo schermo, e poi la segnala all'Arbitro. Può accadere che una Lumaca faccia la prima operazione, e sia poi interrotta da un'altra Lumaca che le porta a termine entrambe consecutivamente. Chi ha letto sia l'articolo del mese scorso che quello di questo mese, può smettere di leggere e cercare di risolvere il problema da solo.

Noi proponiamo una soluzione semplice: la prima Lumaca che arriva al traguardo può impadronirsi dell'Arbitro e "bloccarlo" usandolo come lock. In questo modo le due operazioni diventano un'unica operazione atomica, controllata dall'accesso all'Arbitro:

```
public class Lumaca implements
                         Runnable...
  public void corriFinoAlTraguardo() {
    System.out.println(toString() +
                         ": 1 metro");
    System.out.println(toString() +
                         ": 2 metri");
    synchronized(_arbitro) {
      System.out.println(toString()
                    + ": arrivata!");
      _arbitro.segnalaArrivo(this);}
  }
```

dare una regola semplice: se un oggetto o una variabile è condiviso tra due o più thread, e almeno uno di questi thread ha la possibilità di modificarne il valore, dobbiamo proteggerlo da letture e scritture inconsistenti. Ma come? Nel nostro caso, la risorsa condivisa tra i thread e il Point. Per proteggerlo possiamo usare un vecchio trucco aziendale. In un'azienda dove ho lavorato in passato, le riunioni degeneravano spesso in una grande confusione. Tutti cercavano di interrompersi a vicenda, e chi parlava non riusciva a concludere quel che stava dicendo prima che qualcun altro prendesse la parola.
Esasperati, decidemmo di introdurre una semplice regola: prima delle riunioni si metteva sul tavolo un piccolo oggetto, nel nostro caso un pinguino di peluche. Solo chi aveva il pinguino in mano poteva parlare. Era possibile alzare la mano per chiedere la parola, ma solo il possessore del pinguino poteva decidere di lanciarlo a qualcun altro. Il sistema funzionò perfettamente, e le riunioni diventarono produttive come per incanto. Il sistema dei thread di Java funziona circa nello stesso modo. Quando due thread vogliono accedere ad una risorsa che può servire solo un thread alla volta, il programmatore deve decidere che la risorsa è accessibile solo a chi possiede l'equivalente del pinguino di peluche. In questo caso il pinguino si chiama *lock*. Un thread può mettere le mani sul *lock* solo se nessun altro thread lo ha già preso per sé. Per stabilire quale sia il blocco di codice protetto, e quale sia il *lock*, si usa la parola chiave *synchronized*. I dettagli di questo sistema sono nel box *"Synchronized (rivisitato)"*. Ecco come possiamo proteggere la posizione del Satellite:

```
public class Satellite...
  private synchronized void muovi() {
    verificaCollisioni();
    _posizione.x++;
    _posizione.y++; }
  public synchronized Point getPosizione() {
    return (Point)_posizione.clone(); }
```

Ora tutti i metodi che accedono al campo *_posizione* sono sincronizzati. In questo caso abbiamo sincronizzato i metodi nella loro interezza, quindi come *lock* si usa il Satellite stesso, che ora può arrivare felicemente al termine del suo viaggio:

Numero controlli: 6079
Tempo totale: 58.688

Ho ripetuto il test parecchie volte. Niente errori. Be-

ABBRACCIO MORTALE

I bug nei programmi paralleli sono spesso difficilissimi da trovare. Uno particolarmente insidioso è il cosiddetto *deadlock*, il famigerato "abbraccio mortale" tra thread. Facciamo un esempio:

```
public class Canale {
  private final String _nome;
  public Canale(String nome) {
    _nome = nome;}
  public void comunica(String
                          msg) {
    System.out.println(_nome
              + " -> " + msg);}
}
```

Un *Canale* è un generico canale di comunicazione, che può essere usato da un *Servizio*:

```
public class Servizio implements
                     Runnable {
  private final Canale _canale1;
  private final Canale _canale2;
  private final String _messaggio;
  public Servizio(Canale
      canale1, Canale canale2,
          String messaggio) {
    _canale1 = canale1;
    _canale2 = canale2;
    _messaggio = messaggio;}
  public void run() {
    for(int i = 1; i <= 10; i++) {
      synchronized(_canale1) {
      synchronized(_canale2) {
        _canale1.comunica(
                 _messaggio);
        _canale2.comunica(
                 _messaggio);
      } }
  }  }
}
```

**Un *Servizio* è fatto per girare in parallelo con altri Servizi. Ciascun *Servizio* usa due *Canali* alla volta.
Ora prova a far girare questo semplice programma:**

```
public static void main(String[]
                       args) {
  Canale rete = new Canale(
                      "rete");
  Canale stampante = new
          Canale("stampante");
  Thread s1 = new Thread(new
 Servizio(rete, stampante, "ordi-
                  ne_online"));
  Thread s2 = new Thread(new
      Servizio(stampante, rete,
              "stampa_remota"));
  s1.start();
  s2.start();
}
```

**In tutte le prove che ho fatto sulla mia macchina il programma si blocca quasi immediatamente e senza rimedio, se non quello di terminarlo forzatamente. Il problema è che due *Servizi* partono contemporaneamente e cercano entrambi di acquisire due *lock*, rappresentati dai due *Canali*.
Ma a volte capita che ciascun *Servizio* cerchi di acquisire il *Canale* "lockato" dall'altro *Servizio*. A questo punto nessuno dei due thread riesce a fare alcun progresso, e il programma si ferma. Nel codice che ho scritto per il CD di questo mese ho aggiunto un po' di stampe per mostrare meglio quello che succede:**

- **ordine_online** cerca di acquisire il lock a rete
- **stampa_remota** cerca di acquisire il lock a stampante
- **stampa_remota** ha acquisito il lock a stampante
- **ordine_online** ha acquisito il lock a rete
- **stampa_remota** cerca di acquisire il lock a rete
- **ordine_online** cerca di acquisire il lock a stampante

**Quindi ciascun servizio ha il lock a uno dei due canali e cerca di ottenere il *lock* all'altro. Risultato: il programma è bloccato per sempre.
Nell'esempio qui sopra, il programma si è comportato sempre nello stesso modo in tutte le prove che abbiamo fatto. Nelle situazioni reali, il problema si può manifestare solo una volta ogni tanto, e i thread coinvolti possono essere molti più di due.
Come si può immaginare, problemi come questo possono trasformare in un inferno la vita di un povero sviluppatore. E ora un esercizio: è possibile correggere il programma qui sopra in modo da impedire che si verifichino *deadlock*? Come? Un piccolo aiuto: per farlo basta cambiare una sola riga di codice nel *main()*.**

COSA VUOL DIRE SYNCHRONIZED?

In Java, qualsiasi blocco di codice (cioè qualsiasi sequenza di istruzioni compresa tra due parentesi graffe) può essere sincronizzato. Tutte le sincronizzazioni devono avvenire su un oggetto, che è il *lock* della sincronizzazione. Questo significa che il thread che sta eseguendo quel codice cercherà di ottenere il *lock* appena entra nel blocco, e lo lascerà libero nel momento in cui esce dal blocco. Qualsiasi oggetto Java può essere usato come *lock*.

```
class C {
  public Object o;
  public C(Object
        risorsaCondivisa) {
    o = risorsaCondivisa;}
  public void f() {
    x();
    synchronized(o) {
      y();
      z();}
    w();}
  ... }
```

In questo caso le chiamate ai metodi *y()* e *z()* sono sincronizzate, e il *lock* usato per la sincronia è l'oggetto o. Tutto il gioco della sincronizzazione si basa su una sola regola: solo un thread alla volta può possedere un *lock*. Se un thread cerca di ottenere un *lock* che è già in possesso di un altro oggetto, resta bloccato fino a quando il *lock* non è stato liberato da chi lo possiede. Questo significa che il blocco di codice è diventato atomico, nel senso che non può essere eseguito contemporaneamente da più di un thread. Si dice anche che il blocco sincronizzato è una sezione critica. È possibile anche sincronizzare un intero metodo usando *synchronized* nella sua dichiarazione:

```
class C {
  public synchronized void f2() {
    x();}
  ...
}
```

In questo caso il *lock* è implicito: è l'oggetto sul quale avviene la chiamata al metodo, cioè l'istanza di C sulla quale viene chiamato il metodo *f2()*. In altre parole, questo codice è solo un modo più leggibile per scrivere:

```
class C {
  public void f2() {
    synchronized(this) {
      x(); } }
  ...
}
```

Attenzione: La parola *synchronized* non significa che il codice non può essere chiamato da più di un thread con lo stesso *lock*, ma non impedisce di chiamare il codice con *lock* diversi. È sempre possibile chiamare *f2()* da più thread contemporaneamente, se lo si fa su due istanze diverse di C!

ne! Se la versione precedente del programma non era abbastanza prudente, però, quest'ultima lo è anche troppo. Ogni volta che il metodo *muovi()* viene chiamato, si impadronisce del *lock* sull'istanza del Satellite. Ottenuto il *lock*, il metodo fa i fatti propri, compresa una chiamata a *verificaCollisioni()*. *VerificaCollisioni()* è terribilmente lento, e non ha niente a che vedere con la risorsa condivisa (il campo *_posizione*). Se il *main()* cerca di controllare lo stato del Satellite durante l'esecuzione di *verificaCollisioni()*, resta inutilmente bloccato. Il metodo *muovi()* si comporta in modo maleducato: è come un partecipante alla riunione che si impadronisce del famoso pinguino di peluche e lo tiene per sé mentre chiacchiera al cellulare con la fidanzata. Nel nostro esempio la cosa non crea problemi, ma cosa succederebbe se il client di Satellite fosse il motore grafico del videogioco? Tutte le volte che l'oggetto Satellite resta "lockato", la visualizzazione su schermo si ferma. Questo significa che il videogioco si muove "a scatti". Sarebbe meglio usare il *lock* solo quel tanto che basta per proteggersi da eventuali errori, ma non così tanto da creare ingorghi nell'accesso alle risorse.
Ecco una buona soluzione al problema:

```
public class Satellite...
  private void muovi() {
    verificaCollisioni();
    synchronized(_posizione) {
      _posizione.x++;
      _posizione.y++; }}
  public Point getPosizione() {
    synchronized(_posizione) {
      return (Point)_posizione.clone(); } }
```

Abbiamo cambiato due cose.

Primo: abbiamo sincronizzato solo quei blocchi di codice che scrivono o leggono la risorsa condivisa.
Secondo: anziché usare come *lock* il Satellite, abbiamo usato *_posizione*. Dopo tutto è la *_posizione*, non il Satellite, la risorsa condivisa. Mentre la posizione viene aggiornata, è concepibile che qualcuno voglia accedere ad altri eventuali campi del Satellite. Nel nostro esempio questi altri campi non esistono, ma come regola generale è sempre meglio sincronizzare poco e bene. Ecco il risultato che ho avuto la prima volta che ho fatto girare il programma:

Numero controlli: 10003
Tempo totale: 58.672

Nota che il programma non è più veloce di prima, ma è decisamente più reattivo: il thread del *main()* non deve più fermarsi inutilmente ad aspettare la rilevazione delle collisioni, quindi può fare il suo controllo il doppio delle volte e presumibilmente passa molto meno tempo bloccato in attesa di un *lock*. Questa caratteristica si chiama *liveness*, e fa parte delle prestazioni proprio come la "velocità". Chi scrive un programma concorrente deve continuamente bilanciare le opposte esigenze della correttezza (niente errori sulle risorse condivise) e della liveness (niente attese inutili perché le risorse si liberino). Questi due obiettivi sono solitamente in conflitto, il che spiega perché la programmazione parallela è così difficile. Per questo mese basta così, ma il mese venturo parleremo ancora di thread.
Non mancate!

Paolo Perrotta

Proteggere le immagini

Avete realizzato una bella gallery sul Web e non volete che altri possano appropriarsi indebitamente del vostro lavoro? Ecco come applicare un marchio alle vostre foto

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
C# e concetti base sulla manipolazione di immagini

Software
Framework .NET. Sharp Develop (o .Net SDK o MS Visual Studio .NET)

Impegno

Tempo di realizzazione

Avere un proprio sito web, o alcune pagine web personali, è diventato ormai un'operazione tanto semplice quanto comune, come lo è rendere visibile attraverso quest'ultimo le nostre foto più belle, vista la ormai larghissima diffusione di fotocamere digitali. L'unico problema che si presenta in questi casi è pero inevitabile: pur avendo la paternità delle "opere" che esponiamo, la possibilità che questo diritto non venga rispettato è molto probabile.. ma c'è un modo semplice, se non per impedire, almeno per scoraggiare i più dall'intento di appropriarsi dei diritti sulle nostre immagini.

COSA È UN WATERMARK

Con questo termine (detto anche filigrana) si descrivono delle tecniche per la "protezione" delle immagini. In realtà esistono due tipi di watermark applicabili sulle immagini digitali, quello invisibile e quello visibile. Il primo consiste nell'aggiungere informazioni all'interno dell'immagine (senza che le aggiunte siano visibili in nessun modo nell'immagine stessa) che permettano di risalire all'autore. Il secondo invece, quello trattato in questo articolo, consiste in un vero e proprio overlay sull'immagine che consente di sovrapporre del testo o un'altra immagine a quella originale.
In questo articolo ci occuperemo di questa seconda tecnica, anche se va precisato che il termine esatto di watermarking sottintende un "inserimento" di informazione nell'immagine, mentre nel nostro caso il termine non è proprio adatto visto che si parla di sovrapposizione, ma comunemente viene comunque usato con lo stesso significato.
Sebbene le tecniche che appartengono alla prima categoria sono ovviamente più sicure, anche quelle di watermark visibili sono più che sufficienti, almeno per scoraggiare eventuali violazioni del diritto d'autore. Del resto, come per la protezione del software, se non è possibile evitare con certezza il "furto di immagini", vale comunque la pena di rendere la vita un po' più difficile ai malintenzionati.
Svilupperemo quindi una semplice applicazione .Net che ci permetta, per mezzo dell'utilizzo di GDI+, di ottenere un risultato finale come quello visibile in **Figura 1**.

Fig. 1: *In figura possiamo vedere il risultato finale del nostro lavoro*

Sarà possibile quindi aggiungere alle nostre foto dei watermark sia usando del semplice testo, sia sovrapponendo altre immagini per un effetto "logo" del tutto simile a quello usato per i marchi di numerosi canali televisivi.

INIZIAMO A... RIUTILIZZARE

Andiamo quindi a progettare la nostra applicazione *WindowsForm* che, una volta terminata, sarà esattamente come quella visibile in **Figura 2**.
Come detto in precedenza, questi effetti po-

Fig. 2: *L'applicazione Windows Forms finale in esecuzione*

trebbero essere utili per preparare le foto da caricare su siti web fotografici oppure per aggiungere questa funzionalità ad un nostro sito ASP.NET, per esempio. Alla luce di queste considerazioni implementeremo il codice di elaborazione in una libreria separata, facilmente utilizzabile sia dalla nostra applicazione Windows di esempio, sia da un applicativo web per la gestione di album fotografici che magari svilupperemo in futuro.
La libreria in questione conterrà due metodi distinti che permetteranno di creare watermark a partire da un testo e da una immagine.

AGGIUNGIAMO DEL TESTO

Analizziamo subito il codice per l'inserimento di watermark testuali. Il metodo prende in input un'immagine sorgente ed una stringa, restituendo in output l'immagine di partenza

WATERMARKING IN 6 PASSI CON SHARP DEVELOP

UNA NUOVA APPLICAZIONE

1 Dalla pagina iniziale di SharpDevelop, dopo avere cliccato su *Nuova Combinazione*, apparirà una dialog box. Selezionare un'Applicazione Windows, dargli un nome, e cliccare su *Crea.*

CREAZIONE DELLA LIBRERIA

2 Cliccare sulla combinazione con il tasto destro e selezionare il menu *Aggiunge->Aggiunge nuovo Progetto* per inserire il progetto (Libreria di Classi) per la libreria di watermarking.

AGGIUNTA DEL RIFERIMENTO

3 Clicchiamo col tasto destro su *Refe-renze->Aggiungi Riferimento* della applicazione Windows. Dalla finestra di dialogo apparsa aggiungiamo il riferimento a *ImgLib* dalla sezione *Progetti*.

SCRITTURA DEL CODICE

4 Inseriamo nel file *cs* della libreria il codice dei metodi per l'aggiunta del watermark seguendo quanto esposto nell'articolo. Notate che Sharpdevelop evidenzia la sintassi.

REALIZZAZIONE DELL'INTERFACCIA

5 Realizziamo l'interfaccia dell'applicazione disegnandola con l'editor visuale di SharpDevelop e colleghiamo agli eventi dei bottoni le chiamate ai metodi della libreria.

IL LAVORO COMPLETATO

6 Ciccando su Menu->Esegui (o premendo F5) verrà compilata ed eseguita la nostra nuova applicazione per l'inserimento di watermark nelle immagini.

con il testo “stampato” in sovrapposizione su di essa; La prima istruzione eseguita all’interno della routine sarà la seguente:

```
Graphics grFoto = Graphics.FromImage(imgOriginale);
```

Questo codice si occupa di caricare in un’oggetto *Graphics* l’immagine originale. Questa classe espone gran parte delle funzionalità messe a disposizione da GDI+ per il disegno di primitive in Windows.
Subito dopo troviamo le righe di codice necessarie per la scrittura del messaggio testuale sull’oggetto *Graphics* appena istanziato. In particolare

```
int[] dimFont = new int[]{
                    48,32,24,20,18,16,14,
                              12,10,8,6,4};
Font txtFont = null;
SizeF txtDim = new SizeF();
for (int i=0 ;i<dimFont.Length; i++)
{
   txtFont = new Font("arial", dimFont[i],
                              FontStyle.Bold);
   txtDim = grFoto.MeasureString(testo, txtFont);
   if(txtDim.Width < imgOriginale.Width)
      break;
}
```

Il ciclo *for* determina qual è la dimensione massima del font per la scrittura sull’immagine. Inizialmente viene valutato il font avente size 48 (primo valore dell’array contenente le dimensioni possibili) per poi decrescere sino a trovare il valore che soddisfa la condizione (*txtDim.Width < imgOriginale.Width*), ovvero quel valore che permette di leggere la scritta senza uscire fuori dall’immagine. Tramite il metodo *MeasureString* è infatti possibile stabilire l’ampiezza in pixel di una scritta, data la stringa e le informazioni sul font. Subito dopo il ciclo, sono presenti le istruzioni per posizionare la stringa ad un 5% di distanza dal fondo della foto, con una formattazione centrata.

```
int yPixelDalBasso = (int)(imgOriginale.Height *.05);
float yTesto = ((imgOriginale.Height -
              yPixelDalBasso)-(txtDim.Height/2));

float xTesto = (imgOriginale.Width/2);

StringFormat StrFormat = new StringFormat();
StrFormat.Alignment = StringAlignment.Center;
```

Fatto questo non resta altro che scrivere il testo vero è proprio sull’immagine. Lo faremo in due passaggi per dare un’effetto di ombreggiatura alla scritta finale.

```
SolidBrush brushOmbra = new SolidBrush(
                   Color.FromArgb(153, 0, 0, 0));
grFoto.DrawString(testo, txtFont, brushOmbra,
              new PointF(xTesto+1,yTesto+1),
                                 StrFormat);
SolidBrush brushTesto = new SolidBrush(
           Color.FromArgb(153, 255, 255, 255));
grFoto.DrawString(testo, txtFont, brushTesto,
                  new PointF(xTesto,yTesto),
                                 StrFormat);
grFoto.Dispose();
```

brushTesto e *brushOmbra* sono oggetti di tipo *SolidBrush* che ci permettono di definire il colore della scritta (nel nostro caso il bianco per il testo ed il nero per l’ombra con una trasparenza di circa il 40%) che verranno riversati sull’immagine originale attraverso il metodo *DrawString*. Infine viene invocato il metodo *Dispose* della classe *Graphics* per assicurarsi che vengano rilasciate le risorse GDI+.

E ORA IL LOGO

Il secondo metodo implementato sarà molto simile a quello precedente. Le differenze sostanziali saranno due. La prima per il codice utilizzato per rendere trasparente lo sfondo dell’immagine che verrà usata come logo:

```
ImageAttributes imageAttributes = new
                          ImageAttributes();
ColorMap colorMap = new ColorMap();
colorMap.OldColor = new Bitmap(
             imgWatermark).GetPixel(0,0);
colorMap.NewColor = Color.FromArgb(0, 0, 0, 0);

ColorMap[] remapTab = {colorMap};
```

NOTA

Generalmente in .Net non ci si deve occupare del rilascio delle risorse, in quanto se ne può occupare esclusivamente il Garbage Collector in modo automatico. In alcuni casi particolari però, come per handle di file o come in questo caso oggetti GDI+ (classi che racchiudono risorse non gestite), è opportuno chiamare il metodo *Dispose* per eliminare in modo deterministico le risorse in essi incapsulate.

GDI+

***GDI (Graphics Device Interface)* è una libreria *API (application program interface)*, presente nei sitemi operativi Windows, per l’utilizzo di grafica avanzata bidimensionale nelle applicazioni Windows. Con Windows XP Microsoft ha introdotto la nuova versione (GDI+ appunto), implementata anche nelle classi .NET (quindi usabile in tutti i S.O. Microsoft dove è possibile installare il framework) che consente un’insieme di operazioni più ricca rispetto alla precedente versione, come per esempio la possibilità di effettuare riempimenti sfumati e altri strumenti di disegno avanzati.**

```
imageAttributes.SetRemapTable(
                    remapTab, ColorAdjustType.Bitmap);
```

In particolare viene creato un'oggetto *ColorMap* che permette la rimappatura di colori, indicando come colore di partenza (quello cioè che verrà sostituito) quello del primo pixel in alto a sinistra del logo (si assume che questo pixel farà parte dello sfondo). Come colore di destinazione viene invece specificato un colore completamente trasparente. La mappa di conversione appena impostata verrà usata in seguito, per mezzo dell'oggetto *ImageAttribute* che ne permetterà l'applicazione sull'immagine del watermark.
Il secondo punto cruciale del codice si occupa di rendere il logo semitrasparente prima dell'applicazione:

```
float[][] elementiColorMatrix = {
   new float[] {1.0f, 0.0f, 0.0f, 0.0f, 0.0f},
   new float[] {0.0f, 1.0f, 0.0f, 0.0f, 0.0f},
   new float[] {0.0f, 0.0f, 1.0f, 0.0f, 0.0f},
   new float[] {0.0f, 0.0f, 0.0f, 0.3f, 0.0f},
   new float[] {0.0f, 0.0f, 0.0f, 0.0f, 1.0f}
};

ColorMatrix wmColorMatrix = new ColorMatrix(
                         elementiColorMatrix);
imageAttributes.SetColorMatrix(
          wmColorMatrix, ColorMatrixFlag.Default,
                         ColorAdjustType.Bitmap);
```

Per farlo utilizza una matrice di colore che specifica dei valori di modifica per i colori di ogni singolo pixel di un'immagine. Brevemente permette di specificare nella sua diagonale dei coefficienti di moltiplicazione per le componenti colore dei pixel.
Nella posizione *[0,0]* viene identificato il coefficiente per il valore del rosso (nel nostro caso 1, esattamente come per i valori *[1,1]* e *[2,2]* che si riferiscono rispettivamente a verde e blu), mentre nella posizione *[3,3]* si trova il coefficiente di moltiplicazione per il componente *Alpha*, che rappresenta l'opacità del colore.
Nel nostro caso abbiamo usato un valore di 0.3, che equivale a dire una trasparenza del 70%.
Anche in questo caso l'effetto viene aggiunto all'oggetto *ImageAttribute* precedente che verrà utilizzato nella chiamata successiva di *DrawImage* dell'oggetto *Graphics* che effettuerà il disegno vero e proprio del logo. Inoltre questo metodo include del codice per gestire il posizionamento del logo in base alle scelte dell'utente.

UTILIZZO IN ASP.NET

Implementando i metodi in una libreria distinta, risulta molto semplice riutilizzare queste funzionalità in una applicazione ASP.NET (per esempio un album fotografico web). Per farlo basta aggiungere il riferimento all'assembly realizzato (ImgLib.dll) nel progetto web (esattamente come fatto anche per l'applicazione Windows sviluppata nell'articolo) per poter richiamare le funzioni di watermark dalle nostre pagine web (per esempio dalla pagina web che permette l'upload di una immagine).

Fig. 3: **Applicazione del watermark con immagine con scelta della posizione**

Per il listato completo fare riferimento al codice dell'applicazione incluso nel CD allegato alla rivista.

UTILIZZO DELL'APPLICAZIONE

Una volta in esecuzione il sofware permetterà inizialmente soltanto il caricamento di un'immagine di partenza alla quale applicare un watermark. Una volta fatto questo sarà possibile utilizzare le due versioni implementate per l'inserimento di testo (il quale è impostabile dall'interfaccia attraverso l'apposito campo di edit) e per l'inserimento del logo.
In questo secondo caso sarà anche possibile (vedi **Figura 3**) selezionare dall'apposita combo box la posizione per il logo.

Fabrizio Bernabei

Microsoft .NET Remoting

.NET offre una nuova tecnologia per effettuare l'invocazione remota di oggetti che supera le "ossessioni" e i limiti delle tecnologie precedenti. Introduciamone la teoria e vediamo come usarla

A quanto pare il futuro della programmazione, almeno per quanto riguarda la programmazione aziendale, sta nella capacità di produrre applicazioni distribuite. Un'applicazione distribuita è un progetto costituito da singole parti tali che ciascuna parte potrebbe girare su computer diversi da quello locale.
Ad esempio sul computer A potrebbe girare l'applicazione che recupera certe informazioni da un database, sul computer B potrebbe girare l'applicazione che utilizza questi dati per produrre ad esempio le buste paga, sul computer C potrebbe girare fisicamente il database. Ora, al di la di come questi computer siano fisicamente connessi fra loro, c'è un secondo aspetto da tenere in considerazione, ovvero come le singole applicazioni scambiano i dati fra loro. Il "Remoting" è la soluzione .NET per la costruzione di applicazioni distribuite.
In realtà il "Remoting" in .NET è qualcosa di più, consente infatti non solo di richiamare funzioni esposte da "processi" che girano su macchine remote, ma in modo del tutto trasparente di gestire chiamate a processi differenti che girano sulla stessa macchina.

UN ESEMPIO DI CODICE .NET REMOTING

L'intero articolo si basa sulla creazione di tre componenti.

1) Un oggetto che può essere invocato in modo remoto

2) Un application domain che aspetta le richieste e le gestisce, chiameremo quest'applicazione "Host"

3) Un application domain che effettua le richieste, chiameremo quest'applicazione "Client".

Il nostro scopo sarà quello di gestire la comunicazione fra questi tre componenti utilizzando .NET Remotingi

Fig. 1: Uno schema di come i processi comunicano fra loro

Prima di tutto ci concentreremo sulla realizzazione di un oggetto che esponga metodi che possono essere invocati in modo remoto, considerate il seguente codice:

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ioProgrammo.RemotingServices
{
  public class Server : MarshalByRefObject
  {
    //è stato omessa per brevità tutta la parte di
    //dichiazione di esecuzione del codice di apertura
    //della connessione a SqlServer, di aggancio a
    //Northwind e di e degli Adapter ADO.NET
    //definizione dei Command
```

```
    public Server()
    {
        //
    }
    public string ScalarMethod(string parametro1,
                                   ref int parametro2) {
        parametro2 = parametro2 + 2;
        return "prova: " + parametro1;
    }
    public DataSet GetData() {
        DataSet ds = new DataSet("Ds Prova");
        sqlDataAdapter1.Fill(ds);
        sqlDataAdapter3.Fill(ds);
        sqlDataAdapter2.Fill(ds);
        return ds;
    }
}
```

È quasi bizzarro notare che, a parte l'indizio del dover ereditare dalla classe *MarshalByRefObject*, requisito che comunque non ne preguidica l'esecuzione in locale, non vi è alcun elemento che lascerebbe supporre la natura "remota" di questa classe. C'è un normale costruttore di default, il metodo *ScalarMethod* che accetta parametri scalari una stringa per valore e un intero per referenza e restituisce un altro scalare di tipo stringa. Il metodo *GetData*, invece, restituisce un oggetto strutturato, cioè *DataSet* ed è quindi un esempio interessante di trasferimento di oggetti complessi via remoting. Tale metodo si occupa autonomamente di aprire una connessione ad un Microsoft Sql Server locale puntando al database "didattico" *Northwind.* La connectionstring è letta da una costante presente nella classe. In realtà nella classe sono definiti anche una serie di oggetti ADO.NET che effettivamente effettuano la connessione e che sono stati omessi per brevità. L'esempio completo del codice, l'intera *solution (Remoting.sln),* è disponibile nella directory *Remoting* dei sorgenti allegati. In particolare la classe *Server* è posta in maniera isolata nel progetto *Server**Server.csproj*.

Fig. 2: ***Il file di persistenza di Store visto da un Editor esadecimale***

LA COSTRUZIONE DELL'HOST

Come detto, l'host è un'applicazione che si pone in attesa di richieste per un oggetto e le

CHE COS'È UN APPLICATION DOMAIN?

Tutti sanno che al lancio di un'applicazione segue la creazione di un processo, che altro non è che l'immagine dell'applicazione in memoria. Normalmente lo spazio di memoria riservato ai processi e ai dati che gestiscono è isolato. Il processo *A* non può accedere all'area di memoria riservata al processo *B*.
È anche vero che per scambiare dati fra processi è necessario stabilire un meccanismo di comunicazione fra processi. Normalmente è il sistema operativo tramite l'*RPC – Remote Procedure Call* – a gestire la comunicazione fra processi. Vi sarà spesso capitato di ricevere un errore *"RPC not avalaible"* questo errore viene generato proprio quando il servizio RPC non funzione o non è disponibile, perciò è impossibile comunicare fra processi. .NET supera questo concetto. Di fatto introduce il concetto di *"Application Domain"*. I processi .NET di tipo *"Application Domain"* sono svincolati dall'uso di RPC, la memoria è interamente gestita dal framework di .NET tramite la CLR.
In .NET è possibile creare più *Application Domain,* e persino *Application Domain* annidati all'interno di un'unica applicazione.
Ciascun *Application Domain* gode di una gestione della memoria separata.
Ovviamente *.NET Remoting* deve poi individuare dei metodi affinché gli oggetti appartenenti ai vari *Application Domain* possano comunicare. Se avete fin qui seguito, avrete capito che i processi sono indipendenti fra loro e inaccessibili l'uno con l'altro. Il *Remoting* individua metodi tali che oggetti presenti su *Application Domain* separati possano comunicare.
Quando si prova ad invocare un oggetto presente nell'*Application Domain* corrente, è sufficiente una tradizionale chiamata locale.
Se invece l'oggetto è presente in un diverso *appdomain* si rende necessaria una chiamata remota. In tal caso sul client verrà creato un proxy, cioè un'istanza della classe *TransparentProxy*, mentre sul server viene creato uno stub.
La potenza e la caratteristica innovativa di *.NET Remoting,* rispetto ad altre architetture di remotizzazione concorrenti, è che tutto questo meccanismo è completamente automatico e non è quindi richiesto allo sviluppatore l'onere di codificare tali strati *proxy/stub.* Al punto che, in certi casi, si potrebbe essere persino completamente all'oscuro della natura locale o remota di una chiamata.
.NET si astrae completamente dal concetto di *Application Domain*.
Virtualmente è possibile far dialogare i processi senza avere la minima conoscenza del concetto di *Application Domain.*

gestisce. Adifferenza di altri sistemi di remotizzazione più complessi quali COM+ o CORBA che sono tenuti in piedi da un application server che funge da *Object Broker*, *.NET Remoting* fornisce solo una tecnologia di remotizzazione delle chiamate e non una vera e propria infrastruttura che funga da incubatrice delle chiamate remote. Non esiste un'applicazione standard che possa fungere da Host, dovremo scriverne uno da zero. Se qualcuno a questo punto avrà già pensato di gettare la spugna, si ricreda: scrivere un host è un'operazione talmente banale da risultare imbarazzante. Un host non è altro che una qualsiasi applicazione .NET eseguibile: un'applicazione *WinForms* o *Console* o un *Windows Service*, cioè un buon vecchio servizio NT.

L'unica alternativa alla scrittura di un proprio host è quella di utilizzare *Internet Information Services* come host accettando, però, alcune limitazioni come quella di poter utilizzare solo il *Channel http*.

Scrivere un proprio host, corrisponde a scrivere una, dico una, linea di codice. Il progetto d'esempio è *\remoting\SWFHost\Host .csproj*, che è parte dei sorgenti allegati. Tutto, infatti, si basa sul file di configurazione che, opzionalmente, può accompagnare le applicazioni .NET di tipo *WinForms*, *Console* o *Windows Service*, cioè il file *App.config*, oppure, se si usa IIS come host, il file *Web.config*. L'unico requisito è che l'applicazione, e quindi il processo Windows che l'ha generato, resti in vita per tutto il tempo. Il file può essere aggiunto nel progetto da Visual Studio .NET e, in fase di compilazione, viene copiato nella directory di compilazione del progetto, rinominandolo come l'eseguibile ma con l'estensione finale *.config* (es. *miaApp.exe.config*).

Osserviamo la versione che serve a tenere in piedi il nostro componente da remotizzare *ioProgrammo.RemotingServices.Server:*

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.runtime.remoting>
    <customErrors mode="off" />
    <application name="remotingData">
        <lifetime leaseTime="100D"
        sponsorshipTimeOut="3M"  renewOnCallTime=
                 "5M" leaseManagerPollTime="1M"
        />
    <channels>
    <channel ref="tcp" port="8737">
      <serverProviders>
          <formatter ref="binary" typeFilterLevel="Full"
                          strictBinding="true"/>
      </serverProviders>
    </channel>
    </channels>
    <service>
      <wellknown mode="SingleCall" type=
      "ioProgrammo.RemotingServices.Server, Server"
                         objectUri="myFirstService"
      />
      </service>
    </application>
```

Fig. 3: Un messaggio incapsulato in formato SOAP

Utilizza questo spazio per le tue annotazioni

ARCHEOLOGIA DELLE ARCHITETTURE APPLICATIVE

L'era delle applicazioni gestionali che accedevano fisicamente ai file di database e ai loro indici, che richiedevano la reindicizzazione periodica e che avevano forme di lock primitivi è definitivamente tramontata. Con l'avvento dei database server, potenti motori relazionali a cui si può delegare in toto la gestione fisica dei dati e in cui, attraverso le stored procedure, è possibile persino trasferire porzioni non banali di codice di validazione sul server. Purtroppo, però, questo approccio non risolve il problema di client particolarmente impegnativi per le macchine degli utenti o di situazioni in cui il database server, per ragioni di sicurezza o semplicemente di connettività, non è fisicamente raggiungibile. Sono nate così le cosiddette architetture multi-tier, veri e propri sistemi applicativi distribuiti, cioè strutturate a strati che possono girare su server differenti raggiungibili attraverso protocolli di rete diversificati. Il requisito fondamentale per questo genere di applicazioni è che le varie parti di software e quindi tipicamente i metodi di classi o intere classi siano esposte come servizi sulla rete e richiamabili da un'applicazione client in remoto. Esistono numerose tecnologie proprietarie e standard di questo genere sul mercato, si pensi a J2EE basata su CORBA e Java RMI, a DCOM che è la base di Microsoft COM+ e al protocollo standard SOAP basato su tecnologie povere quali XML e http. Molti di questi protocolli aggiungono un elemento in più: un Object Broker, cioè un componente server attivo ed intelligente che gestisce il traffico delle chiamate, bilancia il carico su più server e crea ambienti protetti di esecuzione delle applicazioni lato server. Si pensi a COM+ o ai vari application server CORBA o J2EE.

```
    </system.runtime.remoting>
</configuration>
```

Chi ha già usato *App.config* per altre ragioni, sa già di cosa si tratta, i pochi altri lo considerino semplicemente una formato standard .NET per dare un file di configurazione alle proprie applicazioni. Banalmente si tratta di un file xml con un nodo principale configuration. Sotto di esso trovano posto diversi altri nodi che servono a configurare aspetti e tecnologie standard del framework: dai file di log, al versioning degli assembly, ecc… E lo si può anche usare per salvare proprie proprietà applicative in luogo dei soliti file *.ini* o di file xml proprietari.
Dunque c'è anche una porzione specifica per remoting, il nodo *system.runtime.remoting*, composta dal nodo *customErrors*, di default impostato a *false*, che indica se deve essere permesso che le eccezioni del componente di remoting risalgano fino al client che l'ha invocato (valore *false*) oppure che tutti gli errori vengano filtrati e sostituiti con messaggi di errore applicativi. A questo punto notiamo un nodo application che rappresenta proprio il nostro servizio di remoting da pubblicare. Può esservi un solo application per host. Nel nostro caso l'applicazione si chiama *remotingData*. In esse sono definiti tutti gli aspetti del servizio esposto: dal channel usato per la comunicazioni, al formatter usato per lo streaming dei dati, al nome della classe vera e proprio da pubblicare.
Il nodo *channels*, innanzitutto, enumera i channel che si intende usare nell'host; nell'esempio viene usato il *TcpChannel* fornito con il .NET Microsoft.
Il servizio verrà pubblicato sulla porta 8737 e farà uso del *formatter* di tipo *binary*, come si evince dal nodo *formatter*.
È interessante rilevare che, sia il formatter che il channel usati, pur essendo delle classi .NET a tutti gli effetti, nel nostro app.config vengono semplicemente identificati rispettivamente con gli alias *"binary"* e *"tcp"*. Ciò è possibile perché la loro defizione completa è presente nel file generale di configurazione del .NET Framework, cioè *machine. config*, che è presente nella directory *<windows_ dir>\Microsoft.NET\Framework\v1.1.4322 \CONFIG* (versione 1.1 del framework).
Ecco, infatti, l'elenco dei channel di "sistema" definiti in questo file:

```
<channels>
    <channel id="http" type="System.Runtime .Remoting
        .Channels.Http.HttpChannel, System.Runtime
        .Remoting, Version=1.0.5000.0, Culture=neutral,
        PublicKeyToken=b77a5c561934e089"/>
    <channel id="tcp" type="System.Runtime.Remoting
        .Channels.Tcp.TcpChannel,
        System.Runtime.Remoting, Version=1.0.5000.0,
        Culture=neutral,
        PublicKeyToken=b77a5c561934e089"/>
</channels>
```

Ragion per cui possiamo riferirci al channel *System.Runtime.Remoting.Channels.Tcp.TcpChannel*, contenuto nell'assembly *System*

IL MARSHALING IN .NET

Ci sono due diversi metodi per rendere disponibile ad un client un oggetto remoto. In generale:

- **un oggetto *Marshal By Value (MBV)* è creato sul server, serializzato in uno stream, trasmesso al client su cui viene ricostruita un'esatta replica dell'oggetto stesso che non ha più alcuna relazione con l'istanza originaria.**
- **un oggetto *Marshal By Reference* (*MBR*), invece, passa il riferimento ad un oggetto creato ed eseguito sul server al client che ne trattiene, dunque, solo un riferimento (*proxy*).**

Di fatto gli oggetti *MBR* si possono considerare i veri oggetti di remotizzazione proprio perché essi effettivamente vengono eseguiti su un server e non semplicemente trasferiti. Inoltre gli *MBV* possono andare incontro ad incongruenze quali il pericolo di portare al loro interno riferimenti a risorse che hanno senso e visibilità solo sul server e che invece li perdono a seguito del trasferimenti. Si pensi, ad esempio, alla connessione ad un database server o l'accesso ad un file fisicamente raggiungibile solo dal server o, peggio, ad una periferica hardware del server. Ad un oggetto MBV è richiesta, come unico requisito, la presenza dell'attributo *Serializable* a livello di classe oppure l'implementazione dell'interfaccia *ISerializable*:

```
[Serializable]
public class MyMBVObject
{
    int Code;
    string Name;
    double Execute(int quantity);
}
//oppure
public class MyMBVObject :
                    ISerializable
{
    int Code;
    string Name;
    double Execute(int quantity);
    //unico metodo da
    //implementare dell'interfaccia
    //ISerializable
    void GetObjectData(
            SerializationInfo info,
        StreamingContext context);
}
```

Un oggetto *MBR*, invece, deve necessariamente discendere direttamente o indirettamente dalla classe *System .MarshalByRefObject:*

```
public class MyMBRObject :
                MarshalByRefObject
{
    int Code;
    string Name;
    double Execute(int quantity);
}
```

.*Runtime.Remoting.dll*, semplicemente con l'alias *tcp*. Ciò non toglie che si possa come definire per stesso, avendo però l'accortezza di usare un nuovo alias per evitarne la duplicazione che, peraltro, solleverebbe un'eccezione a runtime.
Tornando al nostro file *app.config*, tralasciando per questo numero il nodo *lifetime*, giungiamo finalmente al nodo service che rappresenta proprio il nostro servizio esposto. Esso è definito nel sottonodo *wellknown* che riporta, nell'attributo *type*, proprio il *full qualified name* del tipo da pubblicare, cioè il nome della classe comprensivo di namespace (*ioProgrammo.RemotingServices.Server*), e l'assembly in cui è contenuto (*Server.dll*, ma senza l'estensione) semplicemente separati da una virgola.
L'alias simbolico del tipo sarà *myFirstService*, per cui le coordinate complete *(url)* per raggiungere tale servizio saranno, supponendo che l'host giri sulla macchina *MIOSERVER (con ip 192.168 .1 .18)*:

```
tcp://MIOSERVER/remotingData/myFirstService
```

oppure

```
tcp://192.168.1.18/remotingData/myFirstService
```

Tutto ciò che ci resta da fare nell'host è scrivere questa istruzione, magari nella *Form_Load*:

```
System.Runtime.Remoting.RemotingConfiguration
    .Configure(Application.ExecutablePath + ".config");
```

Ed ecco il nostro host bello e pronto! Non ci resta che mandarlo in esecuzione e attendere le richieste del client. Ma quale client? Quello che andremo a scrivere tra poco…

COSTRUIAMO IL CLIENT

A questo punto non vi stupirete dell'estrema banalità dell'implementazione di una chiamata al nostro oggetto *Server* da un client via remoting. Il codice completo del client è *\remoting\Client\Client.csproj*. Anche in questo caso serve un file di configurazione simile a quanto visto per l'host. Eccolo:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.runtime.remoting>
    <application name="myClient">
      <lifetime leaseTime="100D" sponsorshipTimeOut=
                    "3M" renewOnCallTime="5M"
                    leaseManagerPollTime="1M"
      />
      <channels>
        <channel ref="tcp" port="0">
          <clientProviders>
            <formatter ref="binary" strictBinding=
                          "true"/>
          </clientProviders>
        </channel>
      </channels>
     <client>
      <wellknown
          type="ioProgrammo.RemotingServices.Server,
                Server" url="tcp://localhost:8737
                /remotingData /myFirstService"
      />
     </client>
   </application>
  </system.runtime.remoting>
</configuration>
```

All'apparenza le due versioni sono quasi indistinguibili: esiste un nodo client al posto del nodo service, in cui è definito il tipo di cui effettuare l'invocazione remota e il suo url. L'altra minuscola differenza sta nel nodo channel che, oltre all'alias del channel da usare, definisce anche un attributo *port*. Esso non è altro che la parte del client su cui il server potrà aprire una comunicazione in senso contrario (cioè il server che contatta il client) in modo da realizzare una comunicazione bidirezionale.
A differenza dell'host, nel caso del client è richiesta anche l'aggiunta dell'assembly del componente remoto tra i riferimenti del progetto. Osserviamo, infine, come il client effet-

COME COMUNICANO I PROCESSI

Alla base del principio di remotizzazione delle chiamate c'è il meccanismo di serializzazione delle richieste al server e delle risposte e cioè di come vengono salvate le informazioni in un pacchetto informativo dalla struttura nota ai due capi della comunicazione. Tali compito spetta ai Formatter. Il .NET Framework è in questo senso molto flessibile: non prevede solo una modalità di formattazione delle richieste, ma ben due: SoapFormatter che usa il formato XML SOAP per il formato delle chiamate e delle risposte remote e BinaryFormatter che, invece, usa un formato completamente binario e quindi più ottimizzato. Esse non sono che l'implementazione di una specifica più generica e astratta, basata sul polimorfismo di implementazione dell'interfaccia IFormatter, che prevede la libertà si estendere e aggiungere nuovi formati di formattazione semplicemente fornendo nuove implementazioni di tale interfaccia.

tuerà una chiamata ed una esecuzione remota della classe *ioProgrammo.RemotingServices.Server:*

```
System.Runtime.Remoting.RemotingConfiguration
   .Configure(Application.ExecutablePath + ".config");
Server srv = new Server();
DataSet ds = srv.GetData();
```

Non c'è trucco: basta la sola chiamata del caricamento della configurazione di remoting, fatta una sola volta per tutta la durata dell'applicazione e, da quel momento in poi, ogni volta che il client cercherà di invocare la classe *Server,* il sottosistema di remoting di .NET si accorgerà che il tipo *ioProgrammo.RemotingServices.Server* è un tipo remoto raggiungibile all'indirizzo *tcp://localhost/remotingData/myFirstService,* invocherà la chiamata remota, creerà al volo un classe *proxy,* cioè una classe isomorfica, cioè con proprietà e metodi identici a quelli della classe remota, usando il formatter specificato dalla comunicazione preparerà dei pacchetti di dati contenenti i parametri da passare al metodo remoto chiamato, trasferirà l'intera chiamata insieme a questi pacchetti all'oggetto remoto usando il channel previsto per la comunicazione, attenderà il risultato dell'esecuzione e poi effettuerà l'operazione contraria con i valori di ritorno della chiamata indietro verso il client. E tutto in modo completamente trasparente all'utente e, perché no, anche al programmatore! È ancora più straordinario se si pensa che, omettendo l'invocazione del metodo *RemotingConfiguration.Configure,* il codice si trasformerebbe in una banalissima creazione ed invocazione di una classe locale. Infatti, in **Figura 4** è possibile osservare il risultato del *Tooltip Evaluator* di Visual Studio quando si invoca la classe in locale o via remoting.

```
Invocazione locale (nello stesso AppDomain):
     Server srv = new Server();
     DataSet d  srv = {ioProgrammo.RemotingServices.Server}
     EndTime();
     dataGrid1.DataSource = ds;

Invocazione via remoting:
     Server srv = new Server();
     DataSet ds  srv = {System.Runtime.Remoting.Proxies.__TransparentProxy}
     EndTime();
     dataGrid1.DataSource = ds;
```

Fig. 4: ***Invocazione locale e invocazione remota della classe Server***

Come al solito processi anche molto complicati vengono nascosti all'utente dalla potenza del framework. In **Figura 5**, invece, possiamo osservare le due semplici applicazioni host e client in esecuzione. Il client riporta nella taskbar anche i tempi di esecuzione delle chiamate in modo che si possano constatare le differenze.

Fig. 5: ***Il client e l'host in esecuzione***

CONCLUSIONI

Microsoft .NET Remoting è una tecnologia di remotizzazione molto sofisticata e flessibile ma, allo stesso tempo, di facile ed immediato utilizzo. Sebbene sia una soluzione proprietaria e non aderisca a nessuno degli standard di interoperabilità di protocolli remoti, rappresenta di sicuro una validissima soluzione per coloro che vogliano realizzare un'architettura veramente multi-tier (cioè anche in senso fisico e non solo con diversi strati di classi che girano nello stesso eseguibile), ma non vogliano investire risorse per attrezzare server con IIS per far girare i web service o con Object Broker costosi o complessi da gestire. In questo numero abbiamo introdotto la problematica realizzando un semplice esempio di codice. Nel prossimo numero approfondiremo alcuni aspetti più complessi ma più interessanti del protocollo di remoting: vedremo come creare host e invocare oggetti remoti da codice senza l'uso di file di configurazione, come effettuare una chiamata bidirezionale sincrona e asincrona (non bloccante), come evitare di portare sul client l'assembly dell'oggetto server da invocare e come sostituire i channel standard del framework con uno di terze parti pur restando nell'architettura di remoting.

Vito Vessia

Vito Vessia è amministratore e cofondatore della codeBehind S.r.l. http://www.codeBehind.it una software factory di applicazioni enterprise, web e mobile, dove progetta e sviluppa applicazioni e framework in .NET, COM(+) e Delphi occupandosi degli aspetti architetturali. È autore del libro

• PROGRAMMARE IL CELLULARE (Hoepli) 2002

sulla programmazione dei telefoni cellulari connessi al PC con protocollo standard AT+. Può essere contattato tramite e-mail all'indirizzo vvessia@katamail.com.

Grafica ad alte prestazioni

Impareremo qualcosa su SDL, Simple Direct Media Library, una libreria Open Source da trattare con riguardo, per le sue caratteristiche di potenza, affidabilità e semplicità

Utilizza questo spazio per le tue annotazioni

SDL, acronimo di "Simple Directmedia Library", è una libreria C/C++ open source per l'accesso a basso livello a tastiera, video 2D e 3D via OpenGL. È disponibile per numerosi sistemi operativi, Linux e Windows fra tutti, ed esistono molti porting verso altri linguaggi quali: Java, Perl, PHP, Python e Ruby. Queste caratteristiche fanno di SDL un candidato ideale per lo sviluppo di videogame a applicazioni che fanno pesante uso di grafica. Il sito di supporto di SDL, visibile all'indirizzo *www.libsdl.org* offre, oltre alla libreria, la relativa documentazione, applicazioni, newsgroup e tutorial.

SIMULIAMO I PIANETI

Il programma che andremo a sviluppare è

COME INIZIARE

Ci servono pochi strumenti e quasi tutti Open Source. Inoltre tutto funzionerà perfettamente anche su Linux con pochi accorgimenti

1 Prima di tutto è necessario un compilatore C/C++ e, per semplificare l'editing dei sorgenti, un ambiente di sviluppo. Per il compilatore la scelta ricade su MinGW, un porting liberamente disponibile di gcc su Windows in modo da garantire la possibilità a tutti i lettori di seguire e compilare questa demo basata su SDL.

2 Per l'ambiente di sviluppo la scelta ricade su Dev-C++, un IDE anch'esso open source dotato di interessanti caratteristiche quali, fra tutte, il completamento automatico del codice. Tra l'altro Dev-C++ viene distribuito anche in una versione che incorpora MinGW, rendendo il processo di installazione semplice ed immediato.
Trovate DevC++ nel CD allegato alla rivista.

3 Scarichiamo la libreria SDL. Collegandosi all'indirizzo www.libsdl.org. Clicchiamo sul link *sdl1 .2* nella sezione *download*. Nella pagina visualizzata cercate la sezione *development libraries* e scaricate il file *SDL-devel-1.2.8-mingw32 .tar.gz.*

4 Scompattate SDL ed identificate quindi la sotto cartella *include*. Copiate la cartella *SDL* qui contenuta nella sotto cartella *include* di Dev-C++. In questo modo sarà più semplice riferirsi ai file header nei sorgenti. Copiate anche i file contenuti nella cartella *lib* di SDL nella cartella *lib* di Dev-C++.

5 Create ora una cartella dove svilupperemo i sorgenti e copiatevi all'interno il file *SDL .dll* che potete trovare nella cartella *bin* di SDL. Avviate Dev-C++ e create un progetto vuoto. Aprite la voce di menu *project/project option.* In corrispondenza del tab *Parameters*, identificate l'area denominata *linker* e digitate

```
-lmingw32 -lSDLmain -lSDL
```

Questa stringa configura il linker per operare correttamente con SDL.

6 Per far funzionare tutto correttamente è necessario altresì accedere al file *SDL_audio.h* che, avendo seguito i passi sopra indicati, dovrebbe trovarsi, partendo dalla directory di installazione di Dev-C++, nella sotto cartella */include/SDL.*
Portarsi alla linea 97 e cambiare il codice presente

```
void (SDLCALL *filters[10])(struct SDL_AudioCVT
                              *cvt, Uint16 format);
```

in

```
void (*filters[10])(struct SDL_AudioCVT *cvt, Uint16
                                            format);
```

una demo che simula il movimento di un gruppo di pianeti, tracciandone la traiettoria sullo schermo.
Creiamo il nuovo file nella directory del progetto “main.cpp”. Iniziamo importando tutti gli header necessari.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <SDL/SDL.h>
```

Dichiariamo ora alcune costanti e strutture dati che ci saranno utili.
PLANETS è il numero di pianeti di cui visualizzare la traiettoria. *GRAVITATION* è una costante utilizzata per impostare la magnitudine della forza di attrazione tra due pianeti. *X_DIMENSION* e *Y_DIMENSION* sono le dimensioni della finestra all'interno della quale girerà l'applicazione. Mass è una struttura dati che memorizza lo stato di un pianeta. In dettaglio i campi memorizzano la sua massa, le coordinate bidimensionali nello spazio e le componenti orizzontali e verticali della velocità.
Infine è definito un array contenente tutti i pianeti che saranno visualizzati nella demo.

```
const int PLANETS = 15;
const float GRAVITATION = .01;
const int X_DIMENSION = 800;
const int Y_DIMENSION = 600;
const float DT= .1;
struct mass
{
  float m; float x; float y;
  float xspeed; float yspeed;
};
mass planets[PLANETS];
```

DISEGNARE UN PIXEL

La prima funzione che scriveremo è utilizzata per disegnare un pixel sullo schermo. Il pixel viene disegnato esclusivamente se le sue coordinate sono interne alla finestra dell'applicazione, pena il verificarsi di un errore.
SDL_Surface è una struct definita in SDL che rappresenta qualsiasi area che è possibile disegnare; il che include, nel nostro caso, la finestra dell'applicazione. *SDL_MapRGB* converte un colore specificato dalle tre componenti rosso, verde e blu in un intero a 32 bit. La surface è rappresentata in memoria come una sequenza di interi a 32 bit. Ogni 4 byte quindi è definito un pixel. Immaginando di avere impostato una finestra di 320 per 200 pixel a 32 bit di profondità di colore, avremo i primi 320 x 4 = 1280 byte che codificano i pixel della prima riga, i seguenti 1280 byte che codificano i 320 pixel della seconda riga e così via per un totale di 320 x 200 x 4 = 250Kbyte.
Volendo quindi impostare il colore del pixel alle coordinate *x* e *y* si accede prima di tutto al puntatore che punta all'inizio dell'area di memorizzazione dei pixel dello screen.
Si calcola poi quanti byte è lunga una riga dell'immagine. Tale valore viene calcolato utilizzando il membro “*pitch*” che indica la lunghezza in byte di una riga della surface, ricordando che un pixel occupa 4 byte. Con questa informazione si sposta il puntatore in avanti del numero di righe pari alla coordinata *y* e si sposta poi il puntatore ancora in avanti di un numero di pixel pari alla coordinata *x*.
Infine si imposta il colore di tale pixel.

```
void DrawPixel(SDL_Surface *screen, int x, int y,
                       Uint8 R, Uint8 G, Uint8 B)
{
  if(x > 0 && x < X_DIMENSION && y > 0 && y
                       < Y_DIMENSION)
  {
     Uint32 color = SDL_MapRGB(screen->
                       format, R, G, B);
     Uint32 *bufp;
     bufp = (Uint32 *)screen->pixels + y*screen->
                       pitch/4 + x;
     *bufp = color;
  }
}
```

INIZIALIZZARE LE STRUTTURE DATI

Per comodità definiamo la funzione *InitScene()* all'interno della quale inizializzeremo lo stato dei pianeti. Lasciamo come esercizio al

APPLICAZIONI BASATE SU SDL

All'indirizzo http://www.libsdl.org/games.php potete trovare un immediato motore di ricerca che da accesso a numerosi videogame sviluppati con SDL. Oltre a garantire momenti di divertimento possono essere una fonte di ispirazione per i vostri videogame sviluppati con SDL.

SDL E DIRECTX

Nel caso si scegliesse di sviluppare un'applicazione basata su SDL per Windows, verrebbe probabilmente il dubbio del perché utilizzare la libreria open source oggetto dell'articolo e non le DirectX. Sappiate tuttavia che SDL sfrutta, se installate, le librerie DirectX per implementare le funzionalità offerte. In più SDL offre le stesse funzionalità su tutti i sistemi operativi, candidandosi sicuramente come una tra le scelte migliori nel caso si desideri sviluppare un'applicazione multipiattaforma.

lettore quello di scrivere il corpo di questa funzione. Un possibile implementazione potrebbe essere quella di inizializzare lo stato di ogni pianeta nell'array ad un valore casuale in modo che ad ogni avvio la situazione iniziale muti. In linea di massima la posizione dovrebbe essere compresa nelle coordinate visibili definite da *X_DIMENSION* e *Y_DIMENSION*.

```
void  InitScene()
{
   //...
}
```

IL MOVIMENTO

EvolveScene(), partendo dallo stato iniziale dei pianeti ne calcola il seguente. Per fare ciò, per ogni pianeta viene calcolata la forza applicata ad esso dagli altri pianeti nel sistema.

Prima di tutto sono calcolate le distanze sull'asse *x* e *y* tra due pianeti. Tramite il teorema di Pitagora si calcola la distanza effettiva.

Viene poi calcolata la magnitudine della forza tra i due pianeti come proporzionale al prodotto delle masse ed inversamente proporzionale alla distanza al quadrato. Questa formula si rifà ovviamente alla forza di gravità.

Con una semplice proporzione vengono calcolate le componenti *x* e *y* della forza applicate al pianeta partendo dall'intensità della stessa.

Dopo che l'effetto di tutti i pianeti è stato sommato le componenti della forza vengono trasformate in una variazione di velocità riconducendosi alla relazione per cui la variazione di velocità di una massa in un certo intervallo di tempo è proporzionale all'intensità della forza costante applicata durante lo stesso intervallo.

Ovviamente è necessario trovare un compromesso tra l'accuratezza della simulazione equivalente ad un basso valore della costante *DT*, ed una velocità accettabile della stessa, costante *DT* elevata.

```
void EvolveScene()
{
  float fx, fy, f, d, dx, dy;
  for(int i=0; i<PLANETS; i++)
  {
    fx = fy = f = d = dx = dy = 0;
    for(int j=0; j<PLANETS; j++)
    {
      if(j!=i)
      {
        dx = planets[j].x - planets[i].x;
        dy = planets[j].y - planets[i].y;
        d =  sqrt(pow(dx,2) + pow(dy,2));
        f = GRAVITATION * (planets[i].m
                    * planets[j].m) / pow(d,2);
        fx = fx + f * dx/d;
        fy = fy + f * dy/d;
      }
    }
    planets[i].xspeed = planets[i].xspeed + (
                              fx/planets[i].m)*DT;
    planets[i].yspeed = planets[i].yspeed + (
                              fy/planets[i].m)*DT;
    planets[i].x = planets[i].x + planets[i].xspeed;
    planets[i].y = planets[i].y + planets[i].yspeed;
  }
}
```

DISEGNARE I PIANETI

La funzione *DrawScene* invece si occupa di disegnare la scena, attivando un pixel in corrispondenza delle coordinate del pianeta eseguendo un *lock* sulla surface da disegnare.

Difatti è necessario lockare una surface prima di disegnarvi dei pixel. *SDL_LockSurface()* offre proprio questa funzionalità. Ad ogni *lock* deve corrispondere un *unlock*. Questa operazione viene portata a termine dalla funzione *SDL_UnlockSurface()*.

Viene richiamata la funzione *DrawPixel()*, avendo cura di convertire le coordinate di ogni pianeta di tipo *float* in *int*, per rendere compatibile i parametri con la funzione *DrawPixel*.

SDL_Flip() deve essere richiamato per sfruttare il double buffering.

```
void DrawScene(SDL_Surface *screen)
{
  SDL_LockSurface(screen);
  for(int i=0; i<PLANETS; i++)
  {
    DrawPixel(screen, (int)planets[i].x, (int)
                        planets[i].y, 30, 250, 250);
  }
  SDL_UnlockSurface(screen);
```

NOTA

SDL E ALTRE PERIFERICHE

SDL può essere utilizzata non solo per la gestione del video, ma anche di flussi audio e per la gestione del CD-ROM (ad esempio per riprodurre una traccia audio durante lo svolgersi del gioco).Include anche funzioni per la gestione della programmazione multithread ed un modello ad eventi.

SDL È L'UNICA SOLUZIONE ?

SDL non è l'unica libreria open source per lo sviluppo di videogame, facilmente portabile su più piattaforme. Una valida alternativa è la libreria "Allegro". Trovate maggiori informazioni sul sito ufficiale http://www.talula.demon.co.uk/allegro/index.html, ed una serie interminabile di giochi basati proprio su questa libreria all'indirizzo http://www.allegro.cc/

```
  SDL_Flip(screen);
}
```

IL PROGRAMMA

Veniamo ora alla funzione principale che prima di tutto inizializza la libreria SDL. Per fare ciò viene richiamata la funzione *SDL_Init*, passando come argomento la costante *SDL_INIT_VIDEO*.
La funzione restituisce un valore minore di zero nel caso l'inizializzazione del sottosistema specificato non vada a buon fine.

```
int main(int argc, char *argv[])
{
  if ( SDL_Init(SDL_INIT_VIDEO) < 0 )
  {
    printf("Unable to init SDL: %s\n",
                                    SDL_GetError());
    exit(1);   }
  atexit(SDL_Quit);
```

Di seguito viene creato un puntatore alla surface principale. Tale oggetto fornito dalla libreria SDL rappresenta un'area che può sostanzialmente essere disegnata. Le surface possono essere sovrapposte e possono essere copiate zone da una surface all'altra.
La surface viene inizializzata alle dimensioni volute, con profondità di colore a 32. Le costanti *SDL_HWSURFACE* e *SDL_DOUBLEBUF* specificano la creazione di una surface memorizzata nella RAM video con double buffering.

OTTIMIZZIAMO

Doublebuffering è una tecnica che prescrive di utilizzare per la gestione di uno spazio da disegnare due zone di memoria. Mentre la prima zona di memoria memorizza l'immagine correntemente visualizzata, il programma disegna il successivo fotogramma in una seconda area di memoria. Al termine del processo la seconda area di memoria viene visualizzata al posto della prima ed il processo disegna il terzo fotogramma nella prima area di memoria, e così via. Se questa tecnica non fosse utilizzata ma si disegnasse un fotogramma alla volta nella stessa area di memoria l'immagine dovrebbe di volta in volta essere svuotata e ridisegnata, producendo un fastidioso effetto sfarfallio.

```
  SDL_Surface *screen;
  screen=SDL_SetVideoMode(X_DIMENSION,
              Y_DIMENSION, 32,SDL_HWSURFACE|
                                SDL_DOUBLEBUF);
  if ( screen == NULL )
  {
    printf("Unable to set 640x480 video: %s\n",
                                 SDL_GetError());
    exit(1);
  }
```

Realizziamo ora un ciclo che disegna i vari fotogrammi della demo da ripetere sino a quando non viene premuto un tasto.

- **SDL_Event** è una struttura dati definita nella libreria SDL atta a memorizzare quale tasto è stato premuto.
- **SDL_PollEvent()** imposta una variabile *SDL_ Event* con i dati relativi al tasto premuto. *SDL_ QUIT* è il tasto che chiude la finestra, *SDL_KEYDOWN* è la pressione di un tasto, *SDLK_ESCAPE* è il tasto *ESC*. Quando si preme il tasto per chiudere la finestra o *ESC*, l'applicazione termina. Quando viene premuto un qualsiasi altro tasto, i pianeti vengono reinizializzati per un altro disegno.

La funzione *DrawScene()* ad ogni passo per generare il prossimo fotogramma, mentre *EvolveScene()* si occupa di aggiornare le strutture dati.

```
  int done=0;
  InitScene();
  while(done == 0)
  {
    SDL_Event event;
    while ( SDL_PollEvent(&event) )
    {
      if ( event.type == SDL_QUIT ) {done = 1;}
      if ( event.type == SDL_KEYDOWN )
      {
        if ( event.key.keysym.sym == SDLK_ESCAPE )
                                       {done = 1;}
        else{InitScene();
      }
    }
  }
  DrawScene(screen); EvolveScene();
 }
 return 0;
```

Ora è sufficiente compilare ed eseguire il tutto tramite l'opportuno tasto nella toolbar *"compile and run"*.

Daniele De Michelis

SUL WEB

Numerosi sono i tutorial su SDL presenti in rete. Uno dei migliori si trova all'indirizzo http://cone3d.gamedev.net/cgi-bin/index.pl?page=tutorials/gfxsdl/index. Seguendo gli esempi ed il codice sorgente allegato è possibile impadronirsi in breve di tutti i fondamenti della gestione del video con SDL.

Scripting con LUA

Dopo avere visto le caratteristiche interne al linguaggio, analizziamo in questo articolo l'interazione tra script e codice "nativo"
Ecco come interfacciare uno script LUA con un programma C++

LUA è un linguaggio di scripting con diverse caratteristiche molto interessanti. Ad esempio è typeless, cioè non costringe il programmatore a specificare il tipo delle variabili che sta usando (interi, stringhe ecc.). È inoltre scritto completamente in ANSI C, quindi funziona sotto qualsiasi piattaforma che abbia un compilatore C conforme agli standard. Questo garantisce una notevole flessibilità e portabilità. La struttura dati fondamentale in LUA è la tabella. Utilizzando le tabella è possibile fare di tutto: dalla gestione di un semplice array fino alla simulazione di una sorta di programmazione object-oriented. La cosa che però rende LUA estremamente utile ed è, forse, il motivo stesso della sua esistenza è l'integrabilità con il software scritto in altri linguaggi. È possibile infatti utilizzare LUA per "scriptare" un'altro programma, che viene detto programma host, o semplicemente host (ospite). Di seguito vedremo come avviene l'integrazione con programmi scritti in C/C++. Il concetto di base è che l'applicazione host fornisce delle funzionalità allo script LUA (la *host API*). Lo script le utilizza chiamando funzioni in C come se fossero funzioni scritte in LUA. Questo meccanismo consente di separare la parte "tecnica" di un software dalla sua parte "logica". Diventa possibile, in sostanza, modificare radicalmente e in poco tempo la logica di funzionamento dell'host, evitando di incorrere in problemi seri che possano portare a crash dell'applicazione o introdurre insidiosi bug.

CARICARE E ESEGUIRE UNO SCRIPT

LUA funziona col concetto di *"stato"*. Uno *stato* è una struttura dati che contiene le informazioni di esecuzione dell'ambiente run-time di LUA. Ogni stato contiene uno specifico script che è caricato in memoria ed eseguito a richiesta. Per eseguire più di uno script alla volta è quindi sufficiente istanziare due o più stati, assegnando a ognuno lo script che ci interessa. Il seguente programma C++ carica ed esegue lo script chiamato *lua_script .lua:*

```
#include <iostream>
using namespace std;
extern "C"
{  #include <lua.h>
   #include <lualib.h>
   #include <lauxlib.h> }
int main()
{ lua_State* stato = lua_open();
   cout << lua_dofile(stato, "lua_script.lua") << endl;
   lua_close (stato);
   return 0;
}
```

Le prime due righe sono tipiche del C++ e servono a includere la libreria standard di input /output: nulla a che vedere con LUA insomma, ma comunque una cosa essenziale da inserire. Le righe successive sono le direttive di inclusione dei file header tipici di LUA:

- **lua.h**: la libreria principale;
- **lualib.h**: le librerie aggiuntive per la manipolazione di stringhe, per le funzioni matematiche ecc.;
- **lauxlib.h**: la libreria per le funzioni ausiliarie, cioè funzionalità non "di base" ma comunque molto utili in fase di programmazione.

Da notare come le direttive di inclusione siano inserite all'interno della

```
extern "C" { ... }
```

Come detto, infatti, LUA è scritto completamen-

te in C e, compilando in C++, è necessario specificarne la sua "provenienza". Per un linking che vada a buon fine, inoltre, è necessario che il linker "veda" le librerie pre-compilate *lua.lib* e *lualib.lib*. Eseguire uno script è banale: basta ottenere un puntatore a uno stato *lua_State*. Per fare questo si utilizza la funzione *lua_open()*. Successivamente si chiama la funzione *lua_dofile()* passando come parametri lo stato appena creato e il nome delle script da eseguire. Lo script viene eseguito immediatamente se si trova in forma binaria, cioè se è già compilato. Se invece è in forma testuale viene prima compilato e poi eseguito. La *lua_dofile()* restituisce un codice numerico di errore, che il nostro programmino di esempio stampa a schermo tramite *cout*. Se il codice è 0 tutto è andato a buon fine, altrimenti si è verificato qualche problema e LUA sta cercando di segnalarcelo. Una volta utilizzato uno stato è necessario rilasciare le risorse a lui assegnate. Per fare questo si deve chiamare la funzione *lua_close()*. Dopo questa chiamata il puntatore non sarà più valido e risulterà quindi inutilizzabile.

LO STACK

Eseguire uno script LUA da un programma C++ è dunque elementare, tuttavia non è che il primo passo di un lungo viaggio! Provando a fare girare il programma di prima quella che si ottiene è una mesta stampa a schermo di un numero. Se lo script caricato è corretto verrà stampato il valore 0. Tutto quindi è andato bene anche se... non ci siamo accorti di nulla! Lo script è infatti stato eseguito "internamente" al programma, senza fornire risultati di sorta all'esterno. Questo è successo poiché non c'è stato uno scambio di valori tra la parte LUA e quella C++ e il risultato dello script, qualunque esso sia, è rimasto nascosto. Si capisce dunque la necessità di un meccanismo di comunicazione tra LUA e il suo host. La parte LUA dovrà chiamare le funzioni della parte C++ passando loro i parametri corretti. Le funzioni C++ dovranno eseguire il loro compito restituendo un risultato utilizzabile all'interno dello script. Il meccanismo previsto per questo scambio di informazioni è una struttura dati organizzata sottoforma di *"stack"* (pila). Uno *stack* è una struttura dati che prevede l'inserimento e l'estrazione in modalità *LIFO (Last In First Out)*. L'ultimo dato a essere inserito è il primo ad essere estratto. Nello stack di LUA vengono inseriti tutti i riferimenti di interesse: dai dati veri e propri ai riferimenti alle funzioni da chiamare. Per inserire dei valori nello stack è necessario utilizzare le funzioni di tipo *lua_push*()*, alcune delle quali sono:

```
void lua_pushnumber(lua_state* pLuaState,
                                        double dValue);
void lua_pushstring(lua_state* pLuaState, char* pStrValue);
void lua_pushnil(lua_state* pLuaState);
```

Come si può vedere per ogni tipo accettato da LUA (numero, stringa ecc.) esiste la corrispondente funzione C++ da chiamare. Questo perché LUA sarà pure *typeless* ma il C++ di certo non lo è! Da notare inoltre la presenza di una funzione apposita per l'inserimento del valore speciale *"nil"*, l'analogo del *NULL* presente in C++. L'inserimento dei valori nello stack è dunque abbastanza immediato, a differenza di ciò che accade per la loro estrazione. Per "estrazione" si intende la lettura del valore affiorante dallo stack e la sua cancellazione. LUA separa queste due operazioni, per cui fare il classico *"pop"* dello stack consiste in effetti in una sequenza di passi successivi:

1. si ottiene l'indice dell'elemento affiorante nello stack utilizzando la funzione *lua_gettop();*
2. si usa una delle funzioni *lua_to*()* per convertire l'elemento all'indice trovato in precedenza in una variabile C++;
3. si usa la funzione *lua_pop()* per togliere l'elemento affiorante dallo stack.

La cosa migliore è racchiudere queste operazioni ripetitive in una funzione o in una macro, ad esempio:

```
#define PopInt(pLuaState, iVar)  \
{  \
   iVar = (int) lua_tonumber ( pLuaState, lua_gettop(
                                        pLuaState ));   \
   lua_pop( pLuaState, 1); \ }
```

In questo modo potremmo scrivere il seguente codice C++:

```
int X,Y;
X = 0;
Y = 23;
lua_pushnumber ( pLuaState, Y);
cout << "X = " << X << " Y = " << Y << endl;
PopInt( pLuaState, X);
cout << "X = " << X << " Y = " << Y << endl;
```

L'output che otterremo sarà:

```
X = 0 Y = 23
X = 23 Y = 23
```

In pratica abbiamo inserito nello stack il valore di *Y* e successivamente abbiamo fatto il pop assegnando il valore estratto alla *X*.

MULTITHREADING

LUA offre un parziale supporto alla programmazione multithreading. È infatti implementato un sistema di *co-routine* sulla base dei thread. Per creare un nuovo thread in si utilizza la funzione:

```
lua_State
 *lua_newthread
            (lua_State *L);
```

Questa funzione restituisce un puntatore al *lua_State* che rappresenta il nuovo thread. Il nuovo stato condivide con quello originale tutte le variabili globali, come ad esempio le tabelle. Tuttavia ha un suo stack, separato da quello di partenza e indipendente da esso.

ESPORTARE UNA FUNZIONE

Il procedimento di rendere una funzione dell'applicazione host utilizzabile da uno script in LUA è detto *"exporting"*. Per fare l'exporting bisogna semplicemente passare all'ambiente runtime il puntatore alla funzione di interesse insieme a una stringa che ne rappresenterà il nome all'interno dello script. La macro che si occupa di fare questa cosa è la seguente:

```
lua_register ( lua_state* pLuaState, const char*
                    pstrFuncName, lua_CFunction pFunc);
```

La cosa più importante alla quale prestare attenzione, in questa fase, è che le funzioni C++ utilizzabili da LUA devono avere tutte un prototipo ben definito, il seguente:

```
int NomeFunzione (lua_state* pLuaState);
```

in altre parole l'unico parametro passabile è lo stato dell'ambiente LUA ed è obbligatorio che venga restituito un intero. Ma come fare allora se si ha bisogno di passare parametri variegati come stringhe, numeri e così via? È qui che torna utile il discorso precedentemente fatto sullo stack. Tutti i valori dei parametri della funzione che esportiamo andranno messi nello stack e saranno utilizzati nel corpo funzione attraverso le funzioni di manipolazione (push e pop) viste in precedenza. Chiariamo tutto con un esempio. Esporteremo una funzione che ci permette di stampare a schermo del testo stabilito all'interno dello script. In altre parole ci sarà un passaggio di una stringa dall'ambiente LUA a quello C++ e quest'ultimo si occuperà della sua stampa. La funzione è così definita:

```
int StampaTesto( lua_State* pLuaState)
{  //controllo che sia una stringa
   if (!lua_isstring (pLuaState, 1))
   {  lua_error (pLuaState); // Stringa non valida! }
   else
   { cout << "--> " << lua_tostring(pLuaState, 1)
                                          << endl; }
   return 0;
}
```

Questa funzione controlla che l'elemento da stampare sia effettivamente una stringa tramite la funzione *lua_isstring()*. Se il test non va a buon

HOST API

```
// Stampa la stringa passata come primo
                        elemento dello stack
int StampaTesto (lua_State* pLuaState)
// Richiede all'utente un numero compreso tra
                         gli estremi passati
int RichiediNumero (lua_State* pLuaState)
// Genera un numero casuale compreso
                     tra gli estremi passati
int NumeroCasuale (lua_State* pLuaState)
// Inizializza il generatore di numeri random
int InizializzaCasuale (lua_State* pLuaState)
```

1 **Per apprezzare la potenza di LUA realizziamo un piccolo gioco sul lancio dei dadi. Il "motore" sarà in C++, mentre la "logica" verrà implementata con uno script. Vengono riportate le firme delle funzioni esportate.**

LO SCRIPT LUA

```
-- Richiedo il numero di lanci da effettuare
lanci = RichiediNumero(min, MAX);
-- Setup dello stato del gioco
InizializzaCasuale();
punti_gio = 0;
punti_cpu = 0;
```

2 **All'interno dello script possiamo utilizzare le funzioni esportate dal C++. In particolare richiediamo all'utente il numero di lanci da effettuare e inizializziamo generatore random e punteggi.**

I LANCI

```
-- Ciclo dei lanci del dato
for i=1,lanci do

   tiro_gio = NumeroCasuale(1,6);
   StampaTesto("Hai fatto " .. tiro_gio);
   tiro_cpu = NumeroCasuale(1,6);
   StampaTesto("Io ho fatto " .. tiro_cpu);

   -- Stampo l'esito di ogni lancio
   if tiro_gio > tiro_cpu then
      punti_gio = punti_gio + 1;
      StampaTesto("Hai vinto tu :(");
   elseif tiro_gio < tiro_cpu then
      punti_cpu = punti_cpu + 1;
      StampaTesto("Ti ho battuto!");
   else
      StampaTesto("Un pareggio...");
   end

   StampaTesto("");
end
```

3 **Per ogni lancio viene generato casualmente un risultato per il giocatore umano e uno per la CPU. Chi realizza il punteggio maggiore guadagna un punto nel totale del risultato. Con *StampaTesto()* vengono visualizzati i vari lanci.**

RISULTATO FINALE

```
-- Stampo l'esito finale della partita
StampaTesto("");
StampaTesto("TU: " .. punti_gio .. " IO: "
                              .. punti_cpu);
if punti_gio > punti_cpu then
  StampaTesto("Hai vinto la partita! :(");
elseif punti_gio < punti_cpu then
  StampaTesto("Sono un campione!!");
else
  StampaTesto("Un pareggio, la rifacciamo?");
end
```

4 **Alla fine dei lanci decisi dall'utente chi ha totalizzato il punteggio più alto vince. La logica del gioco è molto semplice ma sarebbe stata di difficile lettura se l'avessimo "cablata" all'interno del codice C++.**

UNA PARTITA DI ESEMPIO

```
--> Gioco dei dadi!
--> Scegli quanti lanci effettuare [1-10]
[1-10]: -5
[1-10]: 500
[1-10]: 3
```

5 **Ecco una tipica interazione col nostro programma. Da notare come le eccezioni vengano gestite dal C++, e non dallo script. Ad esempio è impossibile scegliere un numero di lanci negativo o troppo alto.**

fine viene segnalato l'errore utilizzando la funzione *lua_error()*. Il codice C++ che permette di utilizzare la funzione appena definita in uno script LUA è il seguente:

```
int main()
{  lua_State* stato = lua_open();
   lua_register (stato, "StampaTesto", StampaTesto);
   lua_dofile(stato, "lua_script.lua");
   lua_close (stato);
   return 0;}
```

La funzione C++ *StampaTesto()* viene registrata nel *lua_State* "stato" con lo stesso nome *"StampaTesto"*. Subito dopo viene eseguito lo script che la utilizza tramite la *lua_ dofile()*, esattamente come fatto in precedenza. A questo punto possiamo utilizzare la funzione in un nostro script. Ad esempio il seguente codice LUA:

```
StampaTesto ("Esempio di Exporting!");
X = True;
Y = 23;
if X then
   StampaTesto("X e' vera!");
end
StampaTesto("Y vale");
StampaTesto(Y);
```

produrrà questo output:

--> *Esempio di Exporting!*
--> *Y vale*
--> *23*

Al contrario di quanto accaduto nel primissimo esempio di questo articolo, dunque, l'esecuzione dello script ha prodotto un risultato apprezzabile anche al di fuori dell'ambiente di runtime di LUA. Per testare la potenza di questo sistema di scripting si provi ad esempio a modificare lo script LUA. Veder cambiare drasticamente il comportamento del nostro programma C++ senza modificarne una riga di codice è decisamente una bella sensazione! Un esempio ancora più lampante della flessibilità di LUA è fornito dal tutorial in queste pagine.

CONCLUSIONI

Con questo articolo si conclude la breve rassegna su LUA. In questi due appuntamenti abbiamo visto le principali caratteristiche del linguaggio (costrutti, librerie ecc.) nonché il metodo per potere integrare uno script con un programma C++. A questo proposito va anche segnalata la possibilità di chiamare da C++ le funzioni scritte in LUA, cioè il procedimento inverso a quello descritto. Questo è il procedimento, un po' più complesso e scarsamente utilizzato nella pratica, che completa il meccanismo di comunicazione bidirezionale tra LUA e il programma host. Le funzionalità offerte da questo linguaggio sono davvero infinite e il lettore interessato potrà sicuramente cercare ulteriori informazioni al sito ufficiale *www.lua.org*.
Buono scripting!

Alfredo Marroccelli

Integrare il C/C++ con un sistema di scripting come LUA è un lavoro abbastanza semplice ma molto ripetitivo e, pertanto, soggetto a errori indesiderati. L'ideale, in progetti di medio-grandi dimensioni, è quello di affidarsi a un tool semi-automatizzato per la generazione di tutte le corrispondenze tra funzioni C/C++ richiamabili da LUA e viceversa. Uno di questi tool, probabilmente il più utilizzato, si chiama *ToLUA*. *ToLUA* è basato su un file header "ripulito" e genera automaticamente i collegamenti (il "*binding*") tra C/C++ e LUA. Sono mappabili costanti, variabili, funzioni, classi, metodi ecc. Il sito di riferimento è http://www.tecgraf.puc-rio.br/~celes/tolua

LO SVOLGIMENTO

```
--> Hai fatto 1
--> Io ho fatto 1
--> Un pareggio...
-->
--> Hai fatto 6
--> Io ho fatto 1
--> Hai vinto tu :(
-->
--> Hai fatto 5
--> Io ho fatto 2
--> Hai vinto tu :(
-->
--> TU: 2 IO: 0
--> Hai vinto la partita! :(
```

6 **Il ciclo principale dello script viene eseguito dalla macchina virtuale di LUA. Se la logica generale del gioco non dovesse piacerci, potremmo modificarla molto agevolmente cambiando lo script.**

MORRA CINESE

```
-- Stampo l'intestazione del gioco
StampaTesto("Gioco della morra cinese!");
StampaTesto("Scegli tra [1].forbice [2].sasso
                                   [3].carta");

-- Richiedo la scelta del giocatore
tiro_gio = RichiediNumero(min, MAX);

-- Effettuo la scelta della cpu in maniera
                                       casuale
InizializzaCasuale();
tiro_cpu = NumeroCasuale(min,MAX);
```

7 **La potenza di LUA è qui evidente. Cambiando infatti solo lo script, senza toccare il programma host, si ottiene un comportamento totalmente diverso. In particolare programmiamo il celebre gioco della morra cinese.**

FORBICE, SASSO O CARTA?

```
-- Controllo l'esito della "mano"
if tiro_gio == tiro_cpu then
   esito = "Un pareggio...";
elseif (tiro_gio == 1) and (tiro_cpu == 2)
                                          then
   esito = "Ho vinto!";
elseif ... -- Tutte le possibili combinazioni
                                  di risultato

-- Stampo il risultato finale
StampaTesto("Tu hai scelto " .. tiro_gio);
StampaTesto("...contemporaneamente io
                  ho scelto " .. tiro_cpu);
StampaTesto(esito);
```

8 **Viene riportata la parte finale dello script della morra. Entrambi gli script presentati sono disponibili nell'allegato (file dadi.lua e *morra .lua*) insieme all'host in C++ (file LUA2 .cpp).**

Un player mp3 in Java

Alla scoperta di Java Layer una libreria per il supporto mp3. Svilupperemo un semplice player, e lo estenderemo aggiungendo funzionalità interessanti. Infine parleremo di Open Source…

Conoscenze richieste
Linguaggio Java

Software
Java2 SDK 1.3

Impegno
Tempo di realizzazione
Java Layer (*http://www.javazoom.net/javalayer/javalayer.html*) è una libreria che implementa la funzionalità di riproduzione mp3. Supporta i formati *mpeg 1/2/2.5 layer 1/2/3* con un JAR di circa 100KB. È rilasciato sotto licenza LGPL: il suo utilizzo è dunque libero, anche all'interno di applicazioni commerciali. Le API di Java Layer sono un poco differenti a quelle a cui si è abituati utilizzando JMF. Queste sono infatti nate per risolvere esclusivamente il problema della riproduzione di audio mp3. Si noteranno però similitudini con JMF e l'integrazione con JavaSound. L'esempio più semplice che è possibile realizzare con Java Layer utilizza una delle numerose classi di supporto disponibili ed utilizza il metodo *play()* per riprodurre uno specifico file mp3:

```
package net.ioprogrammo.mp3player;
import java.io.IOException;
import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.advanced.jlap;
public class mp3player1 {
    public static void main( String[] args ) throws
                                JavaLayerException,
    IOException {
        jlap player = new jlap();
        player.play("Searching.mp3"); } }
```

in questo esempio viene riprodotto il file di esempio *Searching.mp3*, che ovviamente dovrà essere presente nella directory corrente. Come si nota, le chiamate possono sollevare una eccezione *JavaLayerException*, che identifica problemi vari. Ad esempio, un errore di decompressione mp3. L'eccezione *IOException* indica invece problemi di lettura del file, come il nome errato o un problema con l'accesso al disco.

L'INIZIO E LA FINE

Le API Java Layer offrono alcuni strumenti di controllo dello stato di avanzamento della riproduzione. Ad esempio, è possibile sapere quando inizia o finisce la riproduzione di un file. Per fare questo è possibile registrare un listener di tipo *PlaybackListener.* Per usufruire di questa funzionalità è necessario utilizzare il metodo *playMp3()* della classe *jlap (Java Layer Advanced Player).* Questo si aspetta due

COME INIZIARE

Per eseguire i programmi di esempio è necessario compilare ed eseguire la classe avendo l'accortezza di includere la libreria di Java Layer. Questa è disponibile per il download all'indirizzo citato nell'articolo. Per eseguire il primo esempio seguire i seguenti passi

SCARICAMENTO (OPZIONALE)

1 **Scaricare JavaLayer (il file dovrebbe avere un nome simile a *jLayer1.0 .tar.gz*). al suo interno è presente un file *jar* con le classi. Nel caso della versione 1.0, questo si chiama *jl1.0.jar.* Copiare questo file in un percorso di comodo, come ~/tools/jl. Questo passo è opzionale, in quanto la libreria è già presente nel file degli esempi allegati all'articolo.**

COMPILAZIONE

```
Linux/Mac OS X/Unix $ javac –classpath lib
    /jl1.0.jar –d classes src/net/ioprogrammo
                        /mp3player/mp3player*.java
Windows $ javac –classpath lib\jl1.0.jar
    –d classes src \net\ioprogrammo\mp3player
                                \mp3player*.java
```

2 **Dalla directory dove sono stati scompattati gli esempi, compilare le classi Come in figura**

ESECUZIONE

```
Linux/Mac OS X/Unix
$ javac –classpath lib/jl1.0.jar:classes
    net.ioprogrammo.mp3player.mp3player1
Windows
>javac –classpath lib\jl.0.jar;classes
    net.ioprogrammo.mp3player.mp3player1
```

3 **Per eseguire gli esempi utilizzare il comando come in figura**

parametri: un oggetto *File* che indica cosa riprodurre ed un oggetto *PlaybackListener.*
Questa interfaccia definisce due metodi:

```
public void playbackStarted( PlaybackEvent evt );
public void playbackFinished( PlaybackEvent evt );
```

il primo viene richiamato quando la canzone inizia, il secondo quando finisce. L'evento passato come parametro non fornisce molte informazioni aggiuntive. Il metodo *playMp3()* ritorna un oggetto *AdvancedPlayer* su cui è necessario chiamare il metodo *play()* per avviare la riproduzione:

```
package net.ioprogrammo.mp3player;
import java.io.File;
import java.io.IOException;
import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;
import javazoom.jl.player.advanced.jlap;
public class mp3player2 {
  public static void main( String[] args ) throws
                                   JavaLayerException,
  IOException {
    AdvancedPlayer dp = jlap.playMp3(new File(
                                   "Searching.mp3"),
    new PlaybackListener() {
      public void playbackStarted( PlaybackEvent evt ) {
        System.out.println("partito");}
      public void playbackFinished( PlaybackEvent evt ) {
        System.out.println("finito"); } });
    dp.play(); }
}
```

nell'esempio illustrato vengono stampate semplicemente due scritte, ma in un programma di riproduzione mp3 più completo si potrebbe pensare ad eseguire operazioni più interessanti. Ad esempio, a visualizzare lo stato di *riproduzione/stop* utilizzando icone colorate. Nel caso dell'esempio precedente, il metodo *play()* esegue tutta la canzone, dall'inizio alla fine. Nell'esempio seguente vengono eseguiti solo i primi cinque secondi. Questa prova ci permette di vedere subito la stampa del messaggio *"finito"* associato al listener:

```
package net.ioprogrammo.mp3player;
import java.io.File;
import java.io.IOException;
import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;
import javazoom.jl.player.advanced.jlap;
public class mp3player3 {
  public static void main( String[] args ) throws
                                   JavaLayerException,
  IOException {
    final AdvancedPlayer dp =
     jlap.playMp3(new File("Searching.mp3"),
     new PlaybackListener() {
      public void playbackStarted( PlaybackEvent evt ) {
        System.out.println("partito");}
      public void playbackFinished( PlaybackEvent evt ) {
        System.out.println("finito"); } });
    Thread th = new Thread( new Runnable() {
      public void run() {
        try {
         dp.play(); } catch (JavaLayerException e) {
      e.printStackTrace(); } } });
  th.start();
  try {
     Thread.sleep(5000);} catch (InterruptedException e) {
     e.printStackTrace(); }
   dp.stop();}
}
```

per ottenere una esecuzione di soli cinque secondi non è possibile utilizzare le API standard di Java Layer. Non esiste cioè un metodo che si aspetti il numero di secondi da riprodurre, come ad esempio un *play(int)*. Per simulare questo funzionamento il programma utilizza un thread aggiuntivo. All'interno di questo viene chiamato il metodo *play()*. Nel thread principale viene invece impostata un'attesa di 5000 millisecondi, utilizzando il metodo *Thread.sleep()*.

NOTA

UN FRAMEWORK PER IL MULTIMEDIA

Il supporto più ad alto livello offerto dalla piattaforma Java per la gestione di contenuti multimediali è JMF, la cui homepage è presente all'indirizzo http://java.sun.com/products/javamedia/jmf/index.jsp. Creato in collaborazione con giganti del settore, come Silicon Graphics, estende la piattaforma Java implementando il supporto a diversi formati audio /video.

JAVA NON SUPPORTA NATIVAMENTE MP3

Il formato mp3 è uno dei più diffusi metodi di compressione di dati audio. Abbreviazione di mpeg layer 2, questa tecnologia è utilizzatissima dagli utenti di musica digitale. Oltre ad essere letti da programmi software, i file mp3 possono essere riprodotti sui moderni stereo e su lettori appositi. Inoltre, ultimamente i lettori mp3 stanno sostituendo i lettori CD per la musica portatile. Clamoroso è il successo del lettore iPod di Apple, il lettore di mp3 basato su hard disk più diffuso nel mercato. Il programma iTunes, che lavora in sinergia con iPod, permette di gestire i propri file mp3 e di sentirli sul computer. Questi possono poi essere copiati sull'iPod. All'accoppiata iTunes + iPod è stato affiancato ultimamente anche un negozio online, iTunes Music Store. In questo negozio si possono comprare canzoni singole o interi album, che vengono scaricati direttamente sull'hard disk del computer. Il formato mp3 è dunque fondamentale, ma non è supportato nativamente dalla piattaforma Java. Né le API JavaSound né Java Media Framework lo supportano direttamente. Per poter leggere file mp3 è dunque necessario rivolgersi a librerie aggiuntive come Java Layer.

Quando l'attesa termina, viene chiamato il metodo *stop()*, che interrompe la riproduzione. L'uso di un thread aggiuntivo è necessario perché il metodo *play()* termina solo alla fine della canzone.

UN ACCESSO DIRETTO

Un altro modo per avviare la riproduzione di audio mp3 è quello di accedere direttamente alle classi interne. Che sono poi quelle utilizzate dalla classe jlap per eseguire la riproduzione. Nell'esempio seguente viene utilizzata prima la classe *FileInputStream* per leggere il contenuto del file da riprodurre. Poi viene creato un oggetto *AudioDevice*. Questo modella un dispositivo di riproduzione audio. Poi viene creato un oggetto *Player*, che gestisce la decodifica dei dati mp3 ed il loro invio al dispositivo audio. Per avviare la riproduzione si utilizza come al solito il metodo *play()*.

```
package net.ioprogrammo.mp3player;
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.AudioDevice;
import javazoom.jl.player.FactoryRegistry;
import javazoom.jl.player.Player;
public class mp3player4 {
  public static void main( String[] args ) throws
                                        IOException,
  JavaLayerException {
    InputStream fin = new FileInputStream(
                                  "Searching.mp3");
    BufferedInputStream bin = new
                              BufferedInputStream(fin);
    AudioDevice dev = FactoryRegistry.systemRegistry().
    createAudioDevice();
    final Player player = new Player(bin, dev);
    player.play();}
}
```

le operazioni necessarie all'utilizzo delle classi interne sono poche. La classe *jlap*, che offre una interfaccia ancora più facile, in realtà non è molto complessa. Accedere direttamente alle classi interne offre però delle possibilità in più. Ad esempio, è possibile sapere a che punto è la riproduzione della canzone. In questo modo si può pensare di implementare una sorta di feed-back. Una indicazione all'utente dello stato di avanzamento della riproduzione. Nel codice presente nel listato seguente, è presente una piccola modifica che implementa proprio la stampa del tempo percorso. Questa operazione è affidata ad un nuovo thread, che agisce in parallelo a quello principale. Il thread secondario stampa l'avanzamento, quello principale riproduce l'audio. Nel thread secondario è presente un ciclo infinito. All'interno di questo si ottiene la posizione attuale di riproduzione in millisecondi tramite il metodo *getPosition()*. Con una serie di divisioni intere il numero di millisecondi viene scomposto in ore, minuti e secondi. I dati ottenuti vengono prodotti in output con una formattazione appropriata. L'operazione viene ripetuta ogni 900 millisecondi. Per fare ciò viene utilizzata una chiamata *Thread.sleep()*.

```
package net.ioprogrammo.mp3player;
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.AudioDevice;
import javazoom.jl.player.FactoryRegistry;
import javazoom.jl.player.Player;
public class mp3player4 {
  public static void main( String[] args ) throws
                                        IOException,
  JavaLayerException {
    InputStream fin = new FileInputStream(
                                  "Searching.mp3");
    BufferedInputStream bin = new
                              BufferedInputStream(fin);
    AudioDevice dev = FactoryRegistry.systemRegistry().
                                  createAudioDevice();
    final Player player = new Player(bin, dev);
    Thread th = new Thread( new Runnable() {
      public void run() {
        for(;;) {
          int millisec = player.getPosition();
          int sec = millisec / 1000;
          int minutes = sec / 60;
          int hours = minutes / 60;
          int t_millisec = millisec - sec * 1000;
          int t_sec = sec - minutes * 60;
          int t_minutes = minutes - hours * 60;
          System.out.println( hours + ":" + t_minutes +
                  ":" + t_sec + "." + t_millisec );
          try {
            Thread.sleep(900);
          } catch (InterruptedException e) {
            e.printStackTrace();}}}} );
    th.start();
    player.play(); }
}
```

CREARE UN'INTERFACCIA GRAFICA

A questo punto è possibile pensare di dotare il programma di una interfaccia grafica. In questo momento verrà creata una semplice finestra, dotata so-

NOTA

MP3 PLUGIN

SUN ha rilasciato un plugin specifico per aggiungere il supporto ad mp3 nelle API JMF. Questo è disponibile all'indirizzo http://java.sun.com/products/java-media/jmf/mp3/download.html. Il suo uso è però un po' complicato, perché va installato nelle estensioni e configurato con gli strumenti di JMF. Forse in futuro queste operazioni saranno agevolate con un download unico di JMF che includa anche questa funzionalità.

JAVALAYER J2ME

La piattaforma J2ME offre il supporto ai contenuti multimediali, ma non sempre è possibile riprodurre dati in formato mp3. JavaLayer J2ME vuole rispondere a questa problematica offrendo una piccola libreria (46KB) per questo scopo. La libreria è attualmente in beta e rilasciata sotto LGPL. È supportato solo il formato mpeg 1 layer 3.

Fig. 2: *Il nostro semplice player in fase di riproduzione*

lo dello stato di avanzamento. La finestra è realizzata tramite JFrame, mentre la descrizione in esso contenuta è implementata com JLabel. Questa etichetta possiede un bordo vuoto di 10 pixel, che serve a spaziarla dai bordi. L'aspetto dell'interfaccia è presente in **Figura 2**.

```
package net.ioprogrammo.mp3player;
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.border.EmptyBorder;
import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.AudioDevice;
import javazoom.jl.player.FactoryRegistry;
import javazoom.jl.player.Player;
public class mp3player5 {
   public static void main( String[] args ) throws
                                          IOException,
   JavaLayerException {
      String filename = "Searching.mp3";
      JFrame frame = new JFrame("Eseguo: " + filename);
      final JLabel elapsed = new JLabel("Trascorso:
                                      <indefinito>");
      elapsed.setBorder( new EmptyBorder(10,10,10,10) );
      frame.getContentPane().add(elapsed);
      frame.setLocation(20, 40);
      frame.setSize(250, 50);
      frame.setResizable(false);
      frame.setVisible(true);
      InputStream fin = new FileInputStream(filename);
      BufferedInputStream bin = new
                              BufferedInputStream(fin);
      AudioDevice dev = FactoryRegistry.systemRegistry().
                                   createAudioDevice();
      final Player player = new Player(bin, dev);
      Thread th = new Thread( new Runnable() {
         public void run() {
            for(;;) {
               int millisec = player.getPosition();
               int sec = millisec / 1000;
               int minutes = sec / 60;
               int hours = minutes / 60;
               int t_millisec = millisec - sec * 1000;
               int t_sec = sec - minutes * 60;
               int t_minutes = minutes - hours * 60;
               elapsed.setText( "Trascorso: " + hours +
                        ":" + t_minutes + ":" + t_sec );
               try {
                  Thread.sleep(900);
               } catch (InterruptedException e) {
                  e.printStackTrace();}}
         }} );
      th.start();
      player.play(); }
}
```

la struttura del programma è simile alla versione precedente. Le aggiunte sono all'inizio, dove vengono creati gli oggetti *JFrame* e *JLabel*. L'etichetta viene aggiunta alla finestra utilizzando il metodo *add()*. Con altre chiamate, viene poi impostata la dimensione e posizione della finestra nello schermo, e resa visibile. L'accesso alle classi di riproduzione mp3 è lo stesso dell'esempio precedente, come pure il thread di controllo dell'avanzamento. L'unica differenza, in merito a questa funzionalità, è che la situazione attuale non viene stampata in output. Viene invece costruita una stringa e visualizzata nell'etichetta di testo.

Massimiliano Bigatti

NOTA

FORUM PER L'AUDIO

Un sito molto frequentato che ospita forum su tutte le questioni che ruotano attorno all'audio digitale è http://www.hydrogenaudio.org. I diversi forum parlano di argomenti come codec loseless, hardware, software/ hardware per CD, formati come AAC, MP3, MPC, Ogg Vorbis, streaming, DVD.

LE ALTERNATIVE A JAVA LAYER

Si è visto in queste pagine come iniziare la creazione di un player mp3. Ma sono molti i progetti open source presenti su Internet con lo stesso scopo. Alcuni sono più belli, altri si focalizzano sulle proprie funzionalità.

L'interfaccia spartana di JavaTunes

JAVATUNES

http://www.stigc.dk/projects/JavaTunes/

Questo programma copia in una sua libreria i file mp3, che poi permette di organizzare e riprodurre.

JLGUI

http://www.javazoom.net/jlgui/jlgui.html

È invece un clone interessante di WinAmp, e supporta gli skin nello stesso formato. È sicuramente un progetto interessante e con una interfaccia gradevole. Molti di questi software sono disponibili con licenza open source. Questo consente di studiarne il funzionamento, magari per iniziare il proprio personale progetto. O per collaborare con quello esistente.

jlGui è un clone di WinAmp

Database Access ti ho creato io!

Impariamo come creare un database Access direttamente da .net senza disporre del Programma. Inoltre vi sveliamo il meccanismo della reflection, per non averne più paura

Access, sebbene tenda un po' ad essere messo da parte come contenitore dati per applicazioni da parte di Microsoft che sembra spingere gli sviluppatori verso SQL Server e simili (*MSDE, SQL Server desktop* ecc...), pare abbia ancora un utilizzo molto diffuso tra gli sviluppatori. Certamente Access ha dalla sua parte alcuni notevoli vantaggi:

1. Sta tutto in un file, il che facilita non poco le operazioni di installazione.
2. Ha una buona interfaccia amministrativa, garantita dall'omonimo programma della suite di Office.
3. Per le applicazioni web in hosting presso provider esterni spesso è l'unica soluzione perché nelle offerte di hosting a basso costo SQL Server non è compreso.

Sicuramente, a livello di prestazioni, Access non può competere con altri RDBMS più strutturati ma, diciamolo chiaramente, quante volte ci capita di utilizzare il database per tabelle che contengono solo qualche centinaio di righe? Ed allora, ben venga pure Access! Uno dei motivi per cui Access sia rimasto un po' un "figliastro" di mamma Microsoft è il fatto che nell'implementazione delle librerie ADO di .NET Framework non c'è nessun metodo che consenta di creare un database da programma. Queste funzioni, nel mondo COM, erano affidate alla libreria *ADOX* che, come suggerisce il nome, rappresentano un'estensione di ADO con funzionalità di creazione e gestione di database.

PERCHÉ CREARE UN DATABASE DA PROGRAMMA?

Supponete di avere un'applicazione (*Windows.Forms* o *ASP.NET*) basata su database Access, che debba essere installata senza il vostro intervento, potrebbe essere utile fare in modo che l'applicazione al suo primo avvio controlli se il database esiste o altrimenti lo crei dinamicamente. Per realizzare questo obiettivo, sviluppando un'applicazione .NET, ci possono essere due strade: quella "canonica" e una piccola scorciatoia che potrebbe essere utile in diversi casi.

LA VIA CANONICA

Fig. 1: Aggiunta nuovo riferimento al progetto

Come avrete già intuito si tratta proprio di utilizzare la "vecchia" libreria *ADOX* per fare il lavoro che le librerie *ADO.NET* non fanno: creare un database Access. In primo luogo occorre creare nel progetto un riferimento ad *ADOX*, lo facciamo cliccando con il tasto destro sulla cartella *"references"* del progetto nella finestra *"esplora*

Fig. 2: Scelta della libreria ADOX verso cui creare il riferimento

soluzioni" e, come indicato in **Figura 1**, scegliamo la voce *"aggiungi riferimento"*. Si aprirà quindi una finestra di dialogo che possiamo vedere in **Figura 2** e, tra le voci presenti nel tab *COM*, andremo a scegliere qualcosa che inizi per *"Microsoft ADO Ext."*. Nella macchina possono essere varie versioni della libreria *ADOX;* dovute a services pack, window update ecc... il consiglio è di scegliere le più vecchie (quelle con il numero più basso: 2.7 invece che 2.8 ecc...) per non dover magari vincolare l'utilizzatore del programma ad un aggiornamento.

A questo punto, se tutto è andato per il verso giusto, Visual Studio creerà automaticamente un wrapper alla libreria *ADOX* che verrà presentata nelle *"references"* del progetto come qualsiasi altro assembly .NET (**Figura 3**).

Fig. 3: ***Riferimento creato alla libreria ADOX nel progetto***

Naturalmente dietro c'è un bel po' di complessità che Visual Studio gentilmente ci risparmia comunque, ed è questo che ci interessa, adesso possiamo utilizzare *ADOX* nel nostro progetto.

LE MANI SUL CODICE ...

Una volta che abbiamo creato il riferimento ad *ADOX* nel progetto (per semplicità gli esempi si riferiscono ad un progetto di *Console Application*) vediamo cosa bisogna fare per creare il database in C# e in VB.NET.

Ecco il metodo per creare il database:

C#

```
static void CreaDatabaseAccess(string filename,
                               JetDBTypeEnum dbType)
{
   String cnnString="Provider= Microsoft.Jet.OLEDB.4.0;
      Data source=" + filename + ";Jet OLEDB:Engine
                              Type=" + (int)dbType ;
   ADOX.CatalogClass catalog=new ADOX.CatalogClass();
   catalog.Create(cnnString);
}
```

VB

```
Private Shared Sub CreaDatabaseAccess(ByVal
filename As String, ByVal dbType As JetDBTypeEnum)
   Dim cnnString As String = "Provider=
    Microsoft.Jet.OLEDB.4.0;Data source=" & filename
        & ";Jet OLEDB:Engine Type=" & CInt(dbType)
   Dim catalog As New ADOX.CatalogClass
   catalog.Create(cnnString)
End Sub
```

In pratica questo metodo crea per prima cosa una variabile *String* chiamata *cnnString* che contiene i dati della connessione: il provider, che è sempre l'ormai famoso *Microsoft.Jet .OLEDB.4.0,* il *"Data source"* che è il percorso dove vogliamo che venga creato il file di database ed il tipo di database contenuto in un'enumerazione definita sempre nel progetto:

C#

```
enum JetDBTypeEnum{
   JET_ENGINETYPE_UNKNOWN = 0,
   JET_ENGINETYPE_JET10 = 1,
   JET_ENGINETYPE_JET11 = 2,
   JET_ENGINETYPE_JET20 = 3,
   JET_ENGINETYPE_JET3X = 4, //Access 97
   JET_ENGINETYPE_JET4X = 5 //Access 2000
}
```

VB

```
Public Enum JetDBTypeEnum
   JET_ENGINETYPE_UNKNOWN = 0
   JET_ENGINETYPE_JET10 = 1
   JET_ENGINETYPE_JET11 = 2
   JET_ENGINETYPE_JET20 = 3
   JET_ENGINETYPE_JET3X = 4 'Access 97
   JET_ENGINETYPE_JET4X = 5 'Access 2000
End Enum
```

In realtà abbiamo implementato questa enumerazione solamente per eleganza di scrittura; la stringa di connessione infatti potrebbe essere tranquillamente:

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data source
            <path del db>;Jet OLEDB:Engine Type=5"
```

dove il valore di *Engine Type* se è 5 indica che verrà creato un file compatibile con il formato Access 2000, mentre se è 4 il formato sarà quello di Access 95/97 (naturalmente gli altri valori 1,2,3 si riferiscono a formati di Access ancora precedenti). Ma torniamo al nostro metodo *CreaDatabaseAccess,* la seconda variabile *catalog* instanzia un nuovo oggetto del tipo *ADOX.CatalogClass* del quale successivamente chiama il metodo *Create* passandogli la stringa *cnnString*. Per completare l'opera non resta che chiamare il metodo *CreaDatabaseAccess* nel *Main* dell'applicazione passandogli il path del database:

C#

```
[STAThread]
static void Main(string[] args) {
   string DbPath = System.IO.Path.Combine(
      Environment.CurrentDirectory,"nuovoDb.mdb");
   CreaDatabaseAccess(DbPath,
```

REFLECTION

La Reflection è il meccanismo per recuperare a run time la struttura e gli oggetti contenuti nelle classi e dei moduli. Il .NET Framework mette a disposizione, appunto, l'oggetto System.Reflection che ci consente di accedere alla struttura ed al contenuto dei moduli caricati e di recuperare le risorse contenute in essi.

```
                JetDBTypeEnum.JET_ENGINETYPE_JET4X);
    Console.WriteLine("database creato in {0}",DbPath);
}
```

VB

```
<STAThread()> _
    Shared Sub Main(ByVal args() As String)
        Dim DbPath As String = _ System.IO.Path.Combine(
        Environment.CurrentDirectory, "nuovoDb.mdb")
        CreaDatabaseAccess(DbPath,
                JetDBTypeEnum.JET_ENGINETYPE_JET4X)
        Console.WriteLine("database creato in {0}", DbPath)
    End Sub
```

Si noti che anziché valorizzare la variabile *DbPath* con un percorso assoluto (tipo *"C: \dati \nuovoDb.mdb"* per intendersi) si è usata la directory corrente trovata dinamicamente con la proprietà *Environment.CurrentDirectory* che è stata concatenata al nome del nuovo file attraverso la funzione *Combine* dell'oggetto *System .IO.Path*. Naturalmente non è finita qui! Oltre che creare il database vuoto dovremmo anche metterci dentro qualcosa! Per creare le tabelle, una volta creato il file di database, è sufficiente ricorrere al fido ADO.NET e al linguaggio SQL. Mostriamo di seguito un esempio di metodo per costruire una tabella "utenti" vuota:

C#

```
static void AggiungiTabella(string cnnString){
    OleDbConnection cnn=new OleDbConnection(cnnString);
    string SQL="CREATE TABLE utenti(ID INTEGER
                CONSTRAINT IDK PRIMARY KEY,NOME
                        CHAR,COGNOME CHAR )";
    OleDbCommand cmd = new OleDbCommand(SQL,cnn);
    cnn.Open();
    cmd.ExecuteNonQuery();
    cnn.Close();
}
```

VB

```
Private Shared Sub AggiungiTabella(ByVal cnnString
                                        As String)
    Dim cnn As New OleDbConnection(cnnString)
    Dim SQL As String = "CREATE TABLE utenti(
     ID INTEGER CONSTRAINT IDK PRIMARY KEY,NOME
                        CHAR,COGNOME CHAR )"
    Dim cmd As New OleDbCommand(SQL, cnn)
    cnn.Open()
    cmd.ExecuteNonQuery()
    cnn.Close()
End Sub
```

In pratica basta creare la connessione con la stessa *connectionString* usata per creare il database, e far eseguire ad un oggetto *OleDbCommand* un comando SQL *"CREATE TABLE"*.

L'ALTRA VIA

Non che in quello che abbiamo visto fino adesso ci sia qualcosa che non funziona, ma in realtà è che sono piuttosto allergico all'uso degli oggetti COM in .NET. Il bello del .NET infatti è che, una volta installato il Framework, c'è già tutto il necessario per funzionare: basta copiare l'eseguibile ed è fatta. Se ci mettiamo a utilizzare il COM si ritorna a quello che è stato chiamato il DLL *hell*, l'inferno delle librerie ovvero: versioni differenti da quella che stiamo utilizzando, necessità di aggiornamenti delle macchine ecc... L'altro inconveniente della soluzione vista in precedenza è che la procedura di popolamento del database richiede la ricostruzione di tutta la sua struttura attraverso comandi SQL il che, soprattutto nei casi di database complessi, non è cosa proprio banale. Se il nostro obiettivo è quello di costruire "al volo" un database il Framework offre comunque una scorciatoia interessante. Il concetto è questo: i database Access non corrispondono forse a dei file fisici? Sì. Bene allora perché non includerli nel programma o libreria che stiamo sviluppando (come ad esempio si fa con le icone o le altre risorse). Includere un file in un'Assembly (il prodotto della compilazione, *exe* o *dll*, in .NET si chiama così per chi ancora non lo sapesse) con Visual Studio è una cosa semplicissima: clicchiamo con il tasto destro sull'icona del progetto su *Esplora Soluzioni* e scegliamo *Aggiungi/Aggiungi* elemento esistente aggiungendo il nostro file di Access che avevamo in precedenza preparato. A questo punto quando andremo a ricompilare il progetto il nostro database sarà "inglobato" nell'Assembly.
Si tratta solo di recuperarlo in fase di esecuzione, e qui entra in gioco il meccanismo della *Reflection*.

LA REFLECTION

Attraverso la *Reflection* un assembly è in grado di leggere dei dati in esso contenuti, ma vediamolo praticamente con il metodo *CreaDatabaseAccessDaAssembly* di creazione del database:

C#

```
using System;
using System.Reflection;
using System.IO;
....
static void CreaDatabaseAccessDaAssembly(
                string filename,string resourcename){
Assembly ASM= Assembly.GetExecutingAssembly();
```

LE RISORSE

Le risorse sono file che in fase di compilazione vengono "fusi" con l'eseguibile principale o la libreria principale. Le applicazioni più frequenti di questa tecnica si hanno nell'incorporamento di icone o di stringhe che serviranno in fase di localizzazione del programma ma gli usi a cui si presta sono molteplici. Per incorporare una risorsa (ad esempio un file) in un assembly con il compilatore a riga di comando (vbc per i progetti Visual Basic e csc per C#) occorrerà specificare la direttiva */resource: <file>*. Utilizzando Visual Studio, invece, l'operazione è ancora più semplice: dopo avere incluso un file nel progetto in "proprietà/operazione di generazione" dovremo selezionare il valore "risorsa incorporata".

```
Stream strm=ASM.GetManifestResourceStream(typeof(
                              CreaDatabase),resourcename);
BinaryReader reader=new System.IO.BinaryReader(strm);
FileStream fstrm = File.Create(filename);
BinaryWriter writer=new BinaryWriter(fstrm);
   byte b;
   while(true){
      try {
          b=reader.ReadByte();
          writer.Write(b);}
      catch(System.IO.EndOfStreamException EndEx){
          break;}}
   writer.Close();
   reader.Close();
}
```

VB

```
Imports System
Imports System.Reflection
Imports System.IO
...
Private Shared Sub CreaDatabaseAccessDaAssembly(
          ByVal filename As String, ByVal resourcename
                                              As String)
   Dim ASM As [Assembly] =
                          [Assembly].GetExecutingAssembly
   Dim strm As Stream =
               ASM.GetManifestResourceStream(GetType(
                              CreaDatabase), resourcename)
   Dim reader As New BinaryReader(strm)
   Dim fstrm As FileStream = File.Create(filename)
   Dim writer As New BinaryWriter(fstrm)
   Dim b As Byte
   While (True)
      Try
         b = reader.ReadByte()
         writer.Write(b)
         Catch EndEx As EndOfStreamException
         Exit While
      End Try
    End While
    writer.Close()
    reader.Close()
End Sub
```

La procedura recupera il riferimento all'assembly corrente attraverso il metodo statico *GetExecutingAssembly* dell'oggetto *System.Reflection.Assembly* (si noti che in VB.NET il tipo Assembly si scrive tra parentesi quadre essendo la parola *"Assembly"* tra quelle riservate del linguaggio). Una volta ottenuto il riferimento all'assembly per recuperare il file incorporato in uno *Stream* si utilizza la funzione *GetManifestResourceStream* passandogli come parametri uno dei tipi contenuti nell'assembly ed il nome del file (omettendo il percorso).
A questo punto lo *Stream* ottenuto verrà letto e scritto nel file con una normale operazione di input/output. Naturalmente andrà modificato anche il *Main* del programma scritto in precedenza.

C#

```
[STAThread]
static void Main(string[] args){
   string DbPath = System.IO.Path.Combine(
       Environment.CurrentDirectory, "nuovoDb.mdb");
   CreaDatabaseAccessDaAssembly(
                                DbPath,"nuovoDb.mdb");
}
```

VB

```
<STAThread()> _
    Shared Sub Main(ByVal args() As String)
    Dim DbPath As String = System.IO.Path.Combine(
        Environment.CurrentDirectory, "nuovoDb.mdb")
    CreaDatabaseAccessDaAssembly
                                (DbPath,"nuovoDb.mdb")
    Console.WriteLine("database creato in {0}", DbPath)
End Sub
```

Dove appunto chiamiamo il metodo *CreaDatabaseAccessDaAssembly* passandogli come parametri il percorso del database ed il nome del file incorporato. I vantaggi di questa soluzione sono:

1. non devo creare riferimenti a librerie esterne a .NET
2. non devo ricreare la struttura di tabelle e indici attraverso comandi SQL
3. se cambio la struttura del database è sufficiente una ricompilazione, mentre utilizzando i comandi SQL anch'essi dovrebbero essere aggiornati.

CONCLUSIONI

Abbiamo visto due modi per costruire dinamicamente un database Access: attraverso l'utilizzo della libreria *ADOX* oppure estraendo un file incorporato dall'Assembly.
Utilizzando queste tecniche potremo disporre di uno strumento in più per la gestione dei dati nei nostri programmi .NET garantendo condizioni di avvio certe. Naturalmente gli esempi (in C# ed in VB.NET) sono limitati ed indicativi; potremmo applicare la tecnica a tutti i tipi di progetti: Windows Application, Librerie e, cosa forse ancor più interessante, applicazioni ASP.NET.

Francesco Smelzo

GLOSSARIO

Il termine Interoperabilità riferito al .NET Framework in questo contesto si riferisce alla possibilità di interazione tra codice gestito e codice non gestito (in particolare componenti COM). Per chi fosse interessato ad approfondire l'argomento segnaliamo il link
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconexposingcomcomponentstonetframework.asp

Dentro il sistema con Visual Basic

Molti di voi sapranno certamente che risulta praticamente impossibile accedere alla memoria fisica del proprio sistema attraverso Visual Basic. In realtà, questo non è del tutto vero

In Visual Basic tutti voi ricorderete che per poter leggere il contenuto di un'area di memoria, è sufficiente ricorrere all'API *RtlMoveMemory()*, una funzione che ci consente di poter effettuare la lettura di un numero voluto di byte specificando l'indirizzo di partenza, quello di destinazione ed il numero di byte da "spostare". Solitamente, ci si è affidati ad essa per spostare aree di memoria (apparentemente viste come una normale sequenza di byte), ma che rispecchiavano di fatto delle strutture ben precise, non di rado ottenute partendo da un puntatore al primo byte ritornato da specifiche API di Windows. La sintassi di questa importante funzione, spesso dichiarata con l'alias *CopyMemory()*, è la seguente:

```
Public Declare Sub CopyMemory Lib "kernel32" Alias
    "RtlMoveMemory" (Destination As Any, Source
        As Any, ByVal Length As Long)
```

La parte importante da tener presente sta proprio nel significato dei primi due parametri che identificano rispettivamente l'indirizzo di destinazione e quello sorgente dai quali leggere e copiare il numero di byte voluti. Tutti voi saprete certamente che tali indirizzi sono da ritenere come virtuali, intendendo con questo il fatto che rappresentano indirizzi "relativi" assegnati dal gestore di memoria del proprio sistema al "processo" che ne sta facendo uso. Altrettanto evidente, da quanto appena affermato, è il fatto che essi non corrispondono affatto ad indirizzi assoluti (ossia ai "reali" indirizzi fisici di memoria).
Quello che accade in questi casi è "semplicemente" un'operazione di mapping tra questi indirizzi (visibili dal processo) ed i corrispondenti indirizzi fisici, un'operazione sempre a carico del gestore di memoria. Anche se apparentemente non esiste una chiamata alle API di Windows in grado di "puntare" direttamente ad un indirizzo di memoria fisica attraverso VB, è bene sapere che esiste, all'interno del kernel di Windows NT/2000/XP, un particolare oggetto (di tipo *Section*), messo a disposizione proprio di chi sviluppa device driver e denominato *\Device\PhysicalMemory*. Un oggetto di tipo *Section* rappresenta una porzione di memoria che può essere condivisa ed utilizzata da un processo qualunque per condividere proprie porzioni di memoria con altri processi. Ad ogni section memory possono essere associate una o più viste corrispondenti. Una vista *(view)* altro non è che una porzione di una qualsiasi section attualmente visibile al processo. L'operazione relativa alla creazione di una vista per una determinata sezione è meglio conosciuta con il termine mapping. Ogni processo, a sua volta, può avere a disposizione una o più viste sulla stessa sezione o su sezioni diverse. Tornando quindi all'oggetto *\device\physicalmemory* è sufficiente aggiungere che esso rappresenta proprio la memoria fisica del sistema ed al quale, un qualunque processo (dotato "chiaramente" degli opportuni permessi), può affidarsi per ottenere un accesso in lettura alla memoria fisica. Per molti aspetti, l'utilizzo dell'oggetto *\Device\PhysicalMemory,* non è molto complicato. Per chi non avesse dimestichezza con questo genere di "programmazione", basti ricordare che i passi da svolgere sono

Fig. 1: *In figura è mostrata la situazione della memoria prima e dopo l'utilizzo della API NtMapViewOfSection().*

pochi e semplici:

- ottenere un handle all'oggetto da utilizzare;
- mappare la porzione di memoria fisica in quella virtuale, consentendo al processo di poterla leggere;
- utilizzare i "riferimenti" ottenuti nella maniera che si ritiene opportuna;
- deallocare l'oggetto dopo averlo utilizzato e liberare l'handle.

Vediamo quindi come effettuare ciascuno dei passi poc'anzi menzionati.

OTTENERE L'HANDLE

Il primo passo da compiere può essere portato a termine attraverso l'utilizzo di un'apposita funzione (purtroppo poco documentata) denominata *NtOpenSection()* e presente all'interno della libreria *NTDLL.dll*. La sintassi di questa funzione, all'interno del codice Visual Basic, è la seguente:

Utilizza questo spazio per le tue annotazioni

```
Private Declare Function NtOpenSection Lib "NTDLL.DLL"
 (SectionHandle As Long,ByVal DesiredAccess As Long,
                 ObjAtt As OBJECT_ATTRIBUTES) As Long
```

dove:

- **SectionHandle:** variabile di tipo Long che al termine conterrà l'handle alla section. Questo parametro è passato per riferimento;
- **DesideredAccess:** variabile di tipo *Long* che identifica il tipo di accesso desiderato alla section. Questo parametro è passato per valore.
  I valori possibili possono essere:
  - *SECTION_QUERY*
  - *SECTION_MAP_WRITE*
  - *SECTION_MAP_READ*
  - *SECTION_MAP_EXECUTE*
  - *SECTION_EXTEND_SIZE*
  - *SECTION_ALL_ACCESS*

  Per gli esempi relativi a questo progetto, il valore utilizzato sarà semplicemente *SECTION_MAP_READ*.
- **ObjAtt:** variabile di tipo *OBJECT_ATTRIBUTES* (una struttura che vedremo subito di seguito), passata per riferimento, all'interno della quale sono indicati il nome e gli attributi dell'oggetto section al quale si fa riferimento.

La struttura *OBJECT_ATTRIBUTES* è così costituita:

```
Private Type OBJECT_ATTRIBUTES
    Length As Long
    RootDirectory As Long
    ObjectName As Long
    Attributes As Long
    SecurityDescriptor As Long
    SecurityQualityOfService As Long
End Type
```

Per gli scopi prefissi con questo articolo, è sufficiente tener presente solo alcuni dei parametri di questa struttura, lasciando a zero i restanti. In particolare:

- **Length:** rappresenta la lunghezza della struttura;
- **ObjectName:** rappresenta il puntatore ad una stringa di tipo *UNICODE_STRING* relativa al nome della section da utilizzare ossia *\Device\PhysicalMemory;*
- **Attributes:** combinazione di flag che specificano determinati attributi. Tra i valori possibili, quello espresso dalla costante *OBJ_CASE_INSENSITIVE,* è l'unico utilizzato all'interno del progetto e specifica la modalità (non "case sensitive") utilizzata per confrontare *ObjectName* con i nomi di oggetti già esistenti. Se questo flag non è impostato, tale modalità è determinata dalle impostazioni del sistema.

Il parametro *ObjectName*, come visto poc'anzi, rappresenta il puntatore ad una stringa di tipo *UNICODE_STRING* ossia una struttura così definita:

```
Private Type UNICODE_STRING
  Length As Integer
  MaximumLength As Integer
  Buffer As String
End Type
```

dove:

- **Length:** lunghezza, espressa in byte, della stringa identificata dall'item *Buffer.* Se la stringa è *NULL-terminated,* questo valore non include il carattere nullo finale;
- **MaximumLength:** lunghezza massima dell'item *Buffer* in maniera tale che, se la stringa viene passata a routine di conversione come la *RtlAnsiStringToUnicodeString(),* il valore di ritorno non ecceda la dimensione del buffer;
- **Buffer:** nel nostro caso specifico, è valorizzato con la stringa *\device\physicalmemory.*

A questo punto abbiamo a disposizione tutti i "componenti" che ci occorrono per proseguire questo cammino. Innanzitutto, facendo riferimento al codice allegato, dobbiamo "riempire" le strutture appena dichiarate valorizzando opportunamente i vari item che le compongono. Successivamente, sarà sufficiente una chiamata del tipo:

```
Ret = NtOpenSection(C_HPhysMemory,
                    SECTION_MAP_READ, OA)
```

per ottenere finalmente, all'interno della variabile *C_HPhysMemory*, l'handle all'oggetto section appena aperto.

APRIRE, ALLOCARE E DEALLOCARE

Ora che abbiamo compreso tutti i parametri necessari per richiamare le funzioni *NtOpenSection()* e seguenti, siamo in grado di ottenere l'handle alla section *\Device\PhysicalMemory*. A questo punto non ci resta che compiere il successivo passo di questa fase ossia quello della rimappatura della porzione di memoria fisica in memoria virtuale affinché possa essere letta anche dall'interno di VB. Anche questa volta ci affideremo ad un API importantissima denominata *NtMapViewOfSection()*. La sintassi è la seguente:

```
Private Declare Function NtMapViewOfSection Lib
    "NTDLL.DLL"(ByVal SectionHandle As Long, ByVal
    ProcessHandle As Long,BaseAddress As Long,ByVal
    ZeroBits As Long,ByVal CommitSize As Long,
    SectionOffset As PHYSICAL_ADDRESS,ViewSize As
    Long,ByVal InheritDisposition As Long,ByVal
    AllocationType As Long,ByVal Protect As Long) As
    Long
```

dove

- **SectionHandle:** handle ad un *Section Object;*
- **ProcessHandle:** handle ad un oggetto di tipo *Process*. Nel nostro caso vale *-1* ed identifica il processo corrente;
- **BaseAddress:** puntatore ad un indirizzo virtuale relativo alla memoria mappata. Nel caso questo parametro venga impostata a zero, il sistema si fa carico di ricercare un indirizzo utile per l'allocazione di memoria. Questa scelta è consigliata;
- **ZeroBits:** indica quanti *"high bit"* non devono essere impostati all'interno di *BaseAddress*. Può essere impostato tranquillamente a zero;
- **CommitSize:** dimensione in byte della porzione di memoria che si desidera mappare;
- **SectionOffset:** puntatore all'inizio del blocco di memoria mappato. Rappresenta l'indirizzo di memoria fisica a 64 bit da leggere;
- **ViewSize:** puntatore alla dimensione del blocco mappato (effettivamente letto) espresso in byte. Tale valore è arrotondato sempre alla dimensione della pagina (4096 byte equivalente al valore esadecimale, dichiarato in VB, &H1000). Questo significa che una richiesta di lettura di soli 32 byte, ad esempio, costringerà a caricare (allocare) l'intera pagina all'interno della quale è contenuta la sequenza di questi byte.
- **InheritDisposition:** può assumere valori pari alle costanti *VIEWSHARE* e *VIEWUNMAP*. In particolare:
  - **VIEWSHARE=1** e specifica che la vista creata sull'oggetto *Section* sarà ereditata in ogni processo figlio creato;
  - **VIEWUNMAP=2** e specifica che la vista creata sull'oggetto *Section* non verrà ereditata dai processi figli.
- **AllocationType:** può assumere valore pari a *MEM_COMMIT* o *MEM_RESERVE;*
- **Protect:** imposta i diritti di protezione della pagina. I valori possibili sono definiti dalle seguenti costanti:
  - *PAGE_NOACCESS*
  - *PAGE_READONLY*
  - *PAGE_READWRITE*
  - *PAGE_WRITECOPY*
  - *PAGE_EXECUTE*
  - *PAGE_EXECUTE_READ*
  - *PAGE_EXECUTE_READWRITE*
  - *PAGE_EXECUTE_WRITECOPY*
  - *PAGE_GUARD*
  - *PAGE_NOCACHE*
  - *PAGE_WRITECOMBINE*

  Per gli esempi relativi a questo progetto, il valore utilizzato sarà *PAGE_READONLY.*

A questo punto abbiamo compiuto il passo più importante di questa serie di operazioni. Se tutto è andato nel verso giusto, dovremmo avere a disposizione, in corrispondenza del terzo parametro della *NtMapViewOfSection(), BaseAddress,* l'indirizzo virtuale dal quale ha inizio la sequenza di byte mappati e in *ViewSize* la dimensione della vista sull'oggetto section. Ora siamo finalmente pronti per utilizzare la funzione *CopyMemory()*, che questa volta può essere certamente richiamata con successo poiché sia l'indirizzo sorgente che quello di destinazione, risultano essere entrambi "virtuali". Come anticipato, una volta effettuata l'associazione tra la memoria virtuale e quella fisica e, quindi, dopo aver richiamato la *CopyMemory()*, possiamo eliminare quanto sinora "allocato" attraverso l'apposita API *NtUnmapViewOfSection()*. La sintassi di questa funzione è la seguente:

```
Private Declare Function NtUnmapViewOfSection Lib
    "NTDLL.DLL"(ByVal ProcessHandle As Long,ByVal
    BaseAddress As Long) As Long
```

Il significato dei parametri credo sia abbastanza evi-

Maggiori dettagli relativi all'argomento possono essere reperiti direttamente dal Driver Development Kit (DDK) all'interno del quale sono contenute diverse informazioni importanti sul significato e la sintassi di moltissime API e strutture, comprese quelle utilizzate nel presente articolo.

dente da non richiedere ulteriori spiegazioni. Al termine del suo utilizzo, l'handle deve essere anch'esso liberato. Per fare ciò, è sufficiente affidarsi alla funzione API *CloseHandle()* definita come segue:

```
Declare Function CloseHandle Lib "kernel32" Alias
        "CloseHandle"(ByVal hObject As Long) As Long
```

dove *hObject* è proprio l'handle ottenuto attraverso la *NtOpenSection()* ed identificato dal primo parametro, *SectionHandle*, utilizzato dalla funzione *NtOpenSection()*.

LA CLASSE VISUAL BASIC

Il progetto Visual Basic presentato implementa una semplice classe in grado di realizzare, praticamente, quanto appena menzionato. Esso è costituito semplicemente da due moduli: il "solito" *Generale.bas* ed il modulo di classe *PhysMem.cls*. Il primo racchiude in sé tutte le dichiarazioni utili all'intero progetto, mentre il modulo *.CLS* rappresenta la classe VB vera e propria. In particolare, attraverso quest'ultimo modulo, è possibile istanziare un oggetto di tipo *PhysMem* costituito dalle seguenti proprietà:

- **LowAddress:** rappresenta l'indirizzo fisico da leggere (parte bassa);
- **HighAddress:** rappresenta l'indirizzo fisico da leggere (parte alta);
- **MemLength:** rappresenta la lunghezza dell'area di memoria da leggere;
- **MemReadLength:** rappresenta la lunghezza, espressa sempre in byte, della porzione di memoria realmente letta dalla *NtMapViewOfSection()*. Il valore che può assumere è sempre un multiplo della dimensione della pagina;
- **CodeError:** riporta l'eventuale codice di errore ottenuto attraverso una delle API sopra menzionate. Questa proprietà è di sola lettura;
- **ExitPoint:** codice numerico che aiuta ad identificare l'API che ha prodotto l'errore. I valori possibili possono essere:

  - *0=NtOpenSection();*
  - *1=NtMapViewOfSection();*
  - *2=NtUnmapViewOfSection();*
  - *3=CloseHandle();*

  Questa proprietà è stata introdotta per aiutare l'utente a capire, in caso di errore (proprietà *CodeError* diversa da zero) quale sia stato il punto (l'ultima istruzione) eseguita con successo, permettendogli di regolarsi di conseguenza;
- **PointerMapMem:** rappresenta il puntatore all'area di memoria nella quale saranno trasferiti i byte "fisici" letti;
- **PointerVirtAddr:** identifica l'indirizzo virtuale all'interno del quale sono stati allocati i byte fisici necessari;
- **HPhysMemory:** handle all'oggetto section.

e da soli due metodi:

- **ReadMemPage:** permette di leggere una porzione di memoria fisica, mappandola all'interno di uno spazio di memoria visibile al processo.
- **UnMap:** dealloca l'oggetto section mappato e libera l'handle ad esso.

A questo punto non resta che vedere come sono stati implementati i due metodi (con particolare riferimento al primo), concludendo con un esempio pratico che illustri le impostazioni dei vari parametri. Il metodo *ReadMemPage* è così strutturato:

```
Public Function ReadMemPage() As Long
    Dim Ret As Long
    Dim PhysAddr As PHYSICAL_ADDRESS
    ReadMemPage=0
    ' Imposta l'indirizzo fisico a 64 bit da mappare
      PhysAddr.LowAddr=C_LowAddress
      PhysAddr.HighAddr=C_HighAddress
    ' Imposta la struttura di tipo UNICODE_STRING
      With US
           .Buffer="\device\physicalmemory"&Chr(0)
           .MaximumLength=Len(.Buffer)*2
           .Length=.MaximumLength-2
      End With
    ' Imposta la struttura di tipo OBJECT_ATTRIBUTES
      With OA
           .Length=Len(OA)
           .ObjectName=VarPtr(US)
           .Attributes=OBJ_CASE_INSENSITIVE
           .SecurityDescriptor=0
           .RootDirectory=0
           .SecurityQualityOfService=0
      End With
    ' Ottieni l'handle all'oggetto di tipo Section
Ret=NtOpenSection(C_HPhysMemory,
                                     SECTION_MAP_READ,OA)
    ' Se si riscontrano problemi...
      If Ret Then
         ReadMemPage=Ret
         C_CodeError=Ret
         C_ExitPoint=0
         Exit Function
      End If
    ' Controlla che sia stata impostata la proprietà
                                            MemLength ad
    ' un valore diverso da 0. In caso contrario, imposta
                         la lunghezza dell'area da leggere ad 1
      If C_MemLength <> 0 Then
         C_MemReadLength = C_MemLength
      Else
```

```
        C_MemReadLength = 1
        C_MemLength = 1
    End If
    ' Mappa la memoria fisica con quella virtuale e
            restituisci il puntatore in C_PointerVirtAddr
    Ret=NtMapViewOfSection(C_HPhysMemory,-1&,
        C_PointerVirtAddr,0&,C_MemLength,PhysAddr,
            C_MemReadLength,VIEW_SHARE,0&,
                                PAGE_READONLY)
    ' Se si riscontrano problemi...
    If Ret Then
        ReadMemPage=Ret
        C_CodeError=Ret
        C_ExitPoint=1
        ' ...comunque libera l'handle
          CloseHandle(C_HPhysMemory)
        Exit Function
    End If
    ' Copia i byte mappati all'interno dello spazio di
                    memoria visibile dal processo
    CopyMemory ByVal C_PointerMapMem,ByVal
            C_PointerVirtAddr,C_MemReadLength
End Function
```

Come già spiegato all'inizio, questo metodo compie diverse azioni:

- valorizza opportunamente le strutture di tipo *UNICODE_STRING* e *OBJECT_ATTRIBUTES;*
- richiama l'API *NtOpenSection()* per ottenere l'handle al section object *\device\physicalmemory;*
- effettua l'operazione di mapping ritornando il puntatore all'indirizzo virtuale identificato dalla proprietà *C_PointerVirtAddr;*
- copia la porzione di memoria "ottenuta" all'interno di quella "puntata" da *C_PointerMapMem,* specificata al momento della creazione dell'oggetto.

Per quanto riguarda il metodo *UnMap*, certamente molto più semplice da comprendere, è così strutturato:

```
Public Function UnMap() As Long
    ' Unmapping della section
    Ret=NtUnmapViewOfSection(-1&,C_PointerVirtAddr)
    ' Se si riscotrano problemi...
    If Ret Then
        UnMap=Ret
        CloseHandle C_HPhysMemory
        C_CodeError=Ret
        C_ExitPoint=2
        Exit Function
    End If
    ' Libera l'handle
    Ret=CloseHandle(C_HPhysMemory)
    ' Se si riscontrano problemi
    If Ret=0 Then
        UnMap=Ret
        C_CodeError=Ret
        C_ExitPoint=3
        Exit Function
    End If
End Function
```

Prima di concludere, possiamo mostrare un piccolo esempio che illustra come impostare correttamente i vari parametri dell'oggetto *PhysMem* per leggere una porzione di memoria fisica.

UN SEMPLICE ESEMPIO

Supponiamo di voler leggere il primo byte della BIOS Area, una porzione di memoria sfruttata dal DOS per conservare diverse informazioni utili come porte COM, LPT, stato della tastiera, memoria installata, ecc. Essa, come sarà certamente noto a tutti, è "posizionata" a cominciare dall'indirizzo specificato dalla coppia *"segmento:offset" 0040:0000.*
Per ottenere il risultato voluto, ecco le istruzioni VB necessarie:

```
Dim BiosArea(&H1000) As Byte
Dim PM As New PhysMem
PM.HighAddress=0
PM.LowAddress=&H400
PM.MemLength=&H1
PM.PointerMapMem=VarPtr(BiosArea(0))
` Leggi la memoria fisica (mapping)
  PM.ReadMemPage
` Fa qualcosa con BiosArea[]...
  ...
` Dealloca
  PM.UnMap
```

In corrispondenza di queste istruzioni alcune delle proprietà dell'oggetto istanziato, sono modificate opportunamente. In particolare, i risultati così come verrebbero mostrati da una semplice *Msgbox(),* sarebbero simili ai seguenti:

```
PM.MemLength=1
PM.MemReadLength=4096
PM.PointerVirtAddr=57475072
PM.CodeError=0
PM.ExitPoint=0
```

Si noti, in particolare, che il sistema alloca comunque l'intera pagina (4096 byte) anche se abbiamo richiesto di leggere pochi byte. Ovviamente, la proprietà *PointerVirtAddr,* assumerà un valore diverso di volta in volta, mentre le restanti, in assenza di errore, sono quelle mostrate poc'anzi.

Francesco Lippo

Chiunque decida di approfondire l'argomento sfruttando ad esempio il Device Driver Kit (nel quale è possibile reperire moltissime altre informazioni a riguardo), noterà certamente l'esistenza di "analoghe" funzioni aventi prefisso Zw (ZwOpenSection(), ZwMapViewOfSection(), ecc.). Questo genere di API serve per l'accesso a file ed al registro in modalità KERNEL (ossia ring 0) e risultano implementate all'interno di NTOSKRNL.EXE (com'era facile aspettarsi). Sono inoltre richiamabili soltanto da codice che gira in tale modalità (kernel) e, nel caso specifico, proprio dai device driver. Ci si rende presto conto anche dell'esistenza di altrettanti funzioni all'interno di NTDLL.DLL, apparentemente simili, ma utilizzabili però in modalità USER (ossia ring 3) e di conseguenza richiamabili non solo dai device driver, ma anche dalle applicazioni.

Il JBoss dell'impresa

Impariamo come funziona Jboss, uno strumento indispensabile per lo sviluppo di applicazioni distribuite, sicure, veloci e flessibili Uno standard ormai in ambiente Enterprise

Qualcuno di voi avrà già sentito parlare di J2EE. Si tratta delle specifiche Sun per costruire applicazioni distribuite adatte ad un ambiente business. Per applicazione distribuita si intende un'applicazione tale che le sue varie parti non risiedono su un unico computer ma vengono invece reperite su macchine e sistemi differenti che forniscono servizi e informazioni all'applicazione che le utilizza. Allo stesso modo qualcuno avrà sentito la denominazione di Application Server. Un'Application Server è un contenitore di servizi che vengono esposti e resi disponibili alle applicazioni che ne fanno richiesta. JBoss AS è un Application Server conforme alle specifiche J2EE, è cioè un contenitore di servizi basati su applicazioni Java e resi disponibili in un ambiente distribuito. Nel caso di JBoss alcuni servizi esposti sono, ad esempio, quelli relativi alla gestione della sicurezza in Web Application o anche in applicazioni standalone. Per intenderci una Web Application che volesse accedere ad una risorsa su un computer ne farebbe richiesta a JBoss il quale con metodi interni concederebbe o negherebbe l'accesso alla risorsa. La Web Application può essere ospitata da un Web Container esterno come Tomcat standalone, ma potrebbe anche essere ospitata direttamente all'interno di JBoss AS che infatti integra, al suo interno, un server Tomcat.

COME INIZIARE

Per funzionare, JBoss necessita di un JDK, versione 1.4 o successiva, non basta il JRE. È necessario settare la variabile d'ambiente JAVA_HOME, facendola puntare alla directory di installazione del JDK. L'installazione di JBOSS è semplicissima: è sufficiente decomprimere l'archivio compresso in una directory locale ed... è già finita! Indicheremo con JBOSS_HOME la directory di installazione di JBoss. È sufficiente eseguire il file JBOSS_HOME *\bin\run.bat* in ambiente windows, oppure JBOSS_HOME /bin/run.sh in ambiente Linux, per far partire JBoss. Se tutto è andato a buon fine, puntando il browser su http://localhost:8080 vedrete comparire l'home page di JBoss che mostra alcuni dei servizi offerti. In caso di errore verrà segnalata l'anomalia sulla console.

QUANDO NON SERVE JBOSS E QUAND'È INDISPENSABILE

JBoss non serve se si vuole sviluppare una Web Application che faccia uso solo di Jsp, Servlet e JavaBean. "Non serve" non significa che JBoss non supporti Web Application di questo tipo. Più semplicemente è inutile ricorrere a JBoss in questi casi: è sufficiente un Web Container qual è Tomcat. JBoss, che come si è detto ingloba anche un Tomcat, versione 5 come Web Container, ha un ruolo diverso: quello di fare da contenitore di nuovi e più avanzati servizi e integrarli attraverso le tecnologie definite dallo standard J2EE. La sua infrastruttura serve se si vogliono realizzare applicazioni Enterprise che facciano uso delle sue,numerose,caratteristiche avanzate, prima fra tutte il fatto di possedere un *EJB Contaneir*. Per *EJB Container* si intende proprio il fatto di essere un contenitore di Enterprise Java Beans.
Un EJB è un'applicazione server side che implementa specifici servizi. JBoss in quanto Application Server è un *EJB Container*, contiene cioè numerose applicazioni Java pronte per essere utilizzate in modo distribuito.

LE MANI NEL GIOCATTOLO

È interessante notare la struttura di cartelle creata durante l'installazione. A partire dall'inizio della ge-

REQUISITI

Conoscenze richieste
Principi di J2EE

Software
JBoss, J2EE

Impegno

Tempo di realizzazione

rarchia ovvero *JBOSS_HOME* c'è una serie di sotto cartelle:

La cartella, bin/, è quella in cui si trova lo script di avvio. Al suo interno c'è almeno un altro file importante: *shutdown.bat,* che serve a fermare l'applicazione in esecuzione. Potete avviare shutdown.bat senza nessun parametro, e vi verrà mostrata a video una schermata con un breve elenco di tutti i parametri disponibili.

- **La cartella è client/** contiene tutti i JAR necessari per l'esecuzione dei vari client, la utilizzeremo per compilare l'applicazione di esempio.
- **La cartella lib/** contiene librerie per l'esecuzione dell'Application Server; è raccomandabile non inserirvi librerie proprietarie per preservare la corretta esecuzione del server. L'intero JBoss è configurato all'interno dell'omonima cartella.
- **La cartella server/** contiene varie sotto-cartelle, ciascuna contiene un insieme di file di configurazione, che gestiscono i vari parametri con cui JBoss può essere avviato. I nomi delle sottocartelle sono: *all/, default/, minimal/* e *standard/.* La cartella utilizzata dallo script *run.bat,* quand'è invocato senza parametri, è quella chiamata *"default".* Se se ne vuole utilizzare una diversa, è necessario specificarla a linea di comando dopo aver specificato il parametro *-c* allo script; ecco, per esempio, come eseguire JBoss AS utilizzando la cartella di configurazione *"all"*:

```
run.bat -c all
```

ENTERPRISE JAVA BEAN COSA SONO E COME FUNZIONANO?

Gli Enterprise JavaBean sono la specifica Sun per lo sviluppo di classi che realizzano la logica business in applicazioni Enterprise. In sostanza si tratta di mini-applicazioni java che non hanno alcun compito di output diretto verso l'utente, ma che restituiscono delle informazioni alle applicazioni che le invocano e dialogano con esse utilizzando lo standard J2EE.
Gli EJB sono di tre tipi:

- *Entity Bean*
- *Session Bean*
- *Message-Driven Bean*

Un Entity Bean è permanente, il suo ciclo di vita non è, come tutti gli altri oggetti Java, legato all'applicazione che ne fa uso: essa può essere fermata e fatta ripartire e l'entity bean a cui accedeva continua ad essere quello creato. È accessibile in remoto via rete, nel senso che qualunque applicazione che è in grado di ricercarlo e trovarlo, attraverso appositi meccanismi di ricerca, lo può usare. È condiviso, nel senso che più applicazioni possono richiamarlo contemporaneamente ed ottenere informazioni sincrone alle altre applicazioni che ne fanno uso. La sua esecuzione avviene in remoto, chiamare un suo metodo equivale ad effettuare un'invocazione remota, e attendere il risultato del metodo eseguito sul server che ospita l'*Entity Bean*. La sua "identità" è data da una chiave primaria in pratica è necessario un attributo il cui valore lo identifichi univocamente. È chiaro che per poter essere realizzato un *Entity Bean* necessita di un database o di qualche altra memoria persistente. Un *Session Bean* invece non è permanente, nasce e muore con l'applicazione che lo richiama, chiaramente non è condiviso tra più applicazioni e non mantiene uno "stato persistente" tra una creazione e l'altra. Quando un'applicazione crea un oggetto locale che si riferisce ad un *Session Bean* e invoca uno dei suoi metodi, l'esecuzione del metodo non avviene sull'oggetto creatoin locale, ma viene eseguito in remoto sull'Application Server che espone il bean: in pratica si ottiene un oggetto locale all'applicazione ma l'invocazione dei metodi sull'oggetto locale "scatena" un'elaborazione remota . In questo senso si può dire che i Session Bean vengono utilizzati per rendere distribuita l'elaborazione. Siccome non c'è interesse a condividere gli stessi oggetti (com'è il caso per gli *Entity Bean*) essi non necessitano di chiave primaria o qualsiasi altro riferimento univoco.

```
Infine i Message-Driven Bean si basano su JMS (
                        Java Messaging Services).
```

UN'APPLICAZIONE DI ESEMPIO

Realizzeremo un'applicazione di esempio al fine di mostrare alcune delle funzionalità messe a disposizione da JBoss. Pur nella sua semplicità questa applicazione farà uso di un EJB e di una JSP. In quest'esempio si farà uso unicamente dei tool standard di Java; in particolare si userà il tool "jar" per creare archivi compressi (archivi che, come vedremo, avranno estensione *".jar", ".war"* o *".ear"*- Chi "comanda" il tool jar è uno script scritto per la shell MS-Dos; come pure la compilazione (con il comando javac) e il deploy sarà effettuato attraverso script Ms-Dos; in un ambiente di produzione è molto più agevole utilizzare, per comandare la compilazione, il packaging e il deploy, un tool come *Ant.* Per realizzare un EJB di tipo *session;* è necessario realizzare tre strutture Java:

1. una che implementa la remote interface. Questa interfaccia, semplicemente dichiarerà i metodi

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

che potranno essere richiamati dalle applicazioni che ne vorranno fare uso. Attenzione. In un interfaccia i metodi vengono semplicemente dichiarati, ma non implementati. L'implementazione dei metodi dovrà essere fatta in una classe che usa l'interfaccia.

2. una che implementa la home interface. Tale interfaccia prevede un unico metodo *(create)* per creare un riferimento all'EJB).
3. una classe che è l'implementazione vera e propria. In questa classe implementeremo i metodi che sono stati semplicemente descritti nella remote interface.

L'esempio creerà un EJB che espone un metodo che restituisce un numero. La realizzazione è volutamente semplice: il numero restituito dal metodo sarà sempre uguale. In un'implementazione "reale" è verosimile che i dati siano dinamici e reperiti da un database, sul file system o attraverso qualche altro mezzo, per esempio via SMTP o via JNDI. Lo scopo è quello di mostrare l'architettura non fornire una realizzazione di un'applicazione Java completa. Realizzeremo le tre classi di cui sopra, all'interno del package *it.ioprogrammo.ejbTest.* I file che le conterranno, saranno rispettivamente:

1. *SessionEJB.java*
2. *SessionEJBHome.java*
3. *SessionEJBBean.java*

La remote dichiara i metodi esposti verso chi vuol far uso dell'EJB; in questo caso *SessionEJB.java* contiene unicamente un metodo:

```
package it.ioprogrammo.ejbTest;
import java.lang.*;
import java.rmi.RemoteException;
import javax.ejb.*;
public interface SessionEJB extends javax.ejb.EJBObject{
  public String getUltimaRivista() throws
                              java.rmi.RemoteException;}
```

La *Home* fornisce un unico metodo: *create()*. Attraverso l'invocazione dello stesso si ottiene un'istanza locale dell'EJB (si ricorda che un SessionBean prevede la creazione di un oggetto locale, fatta per l'appunto invocando questo metodo) su cui eseguire l'invocazione dei suoi metodi di business (la cui esecuzione, come già detto, avverrà in remoto all'interno dell'Application Server); *SessionEJBHome* contiene il seguente codice:

```
package it.ioprogrammo.ejbTest;
import java.lang.*;
import java.rmi.RemoteException;
import javax.ejb.*;
public interface SessionEJBHome extends
                                        javax.ejb.EJBHome{
  public it.ioprogrammo.ejbTest.SessionEJB
    create() throws javax.ejb.CreateException,
                                java.rmi.RemoteException;
}
```

Infine *SessionEJBBean* realizzerà il metodo *getUltimaRivista(),* dichiarato nell'interfaccia *SessionEJB* e, in più, dovrà dichiarare tutti i metodi esposti dall'interfaccia *SessionBean:*

```
package it.ioprogrammo.ejbTest;
import java.rmi.RemoteException;
import javax.ejb.*;
public class SessionEJBBean implements SessionBean{
  public String getUltimaRivista(){
    return "92"; }
  public void ejbCreate() throws CreateException {}
  public void setSessionContext( SessionContext
                        aContext ) throws EJBException {}
  public void ejbActivate() throws EJBException {}
  public void ejbPassivate() throws EJBException {}
  public void ejbRemove() throws EJBException {}
}
```

COMPILARE L'EJB

Una volta scritti i file Java che realizzano l'EJB, è necessario compilarli; ecco un semplice script shell di Windows che lo fa. Si noti come si utilizza il file di JBoss *\client\jbossall-client.jar* per reperire le librerie necessarie:

```
set JBOSS_HOME=c:\path\jboss-4.0.1RC1
set CLIENT_JAR=%JBOSS_HOME%\client
                                \jbossall-client.jar
%JAVA_HOME%\bin\javac -classpath %CLIENT_JAR%
  it/ioprogrammo/ejbTest/SessionEJBBean.java
%JAVA_HOME%\bin\javac -classpath %CLIENT_JAR%
  it/ioprogrammo/ejbTest/SessionEJB.java
%JAVA_HOME%\bin\javac -classpath %CLIENT_JAR%;.
  it/ioprogrammo/ejbTest/SessionEJBHome.java
```

EFFETTUARE IL DEPLOY

Una volta effettuata la compilazione si hanno a disposizione i file *.class.* Per effettuare il deploy è necessario impacchettarli in un file JAR e includere in questo JAR degli opportuni descrittori. JBoss necessita di due file descrittori. Per l'esempio che si è realizzato il primo file si chiama *ejb-jar.xml* e contiene:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC
"-//Sun Microsystems, Inc.//DTD Enterprise
                                JavaBeans 2.0//EN"
```

JBOSS HISTORY

La prima versione dell'Application Server di JBoss, abbreviato JBoss AS nasce nel '99. Ad oggi è uno dei prodotti più famosi distribuiti con licenza Open Source; deve la sua fama al livello di assoluto valore e al fatto che chi lo produce offre supporto di qualità elevata, anche se a pagamento. Il suo utilizzo è particolarmente adatto, per le funzionalità che espone, a tutti quegli ambienti "mission critical" tipici delle realtà commerciali. Il sito di riferimento del prodotto è http://www.jboss.org/products/jbossas. Da tale sito è possibile eseguire il download di JBoss AS sia in forma binaria sia in forma sorgente. In questo articolo si fa riferimento alla versione 4.0.1 di JBoss, che al momento è anche l'ultima versione disponibile.

```
"http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
<description>test di un SessionEJB</description>
<display-name>SessionEJB</display-name>
<enterprise-beans>
  <session id="test_SessionEJB">
  <display-name>SessionEJB</display-name>
  <ejb-name>SessionEJB</ejb-name>
<home>it.ioprogrammo.ejbTest.SessionEJBHome</home>
  <remote>it.ioprogrammo.ejbTest.SessionEJB</remote>
  <ejb-class>it.ioprogrammo.ejbTest.SessionEJBBean
                                        </ejb-class>
  <session-type>Stateless</session-type>
  <transaction-type>Container</transaction-type>
  </session>
</enterprise-beans>
<assembly-descriptor>
</assembly-descriptor>
</ejb-jar>
```

Il secondo file, specifico per JBoss, si chiama jboss .xml e contiene:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS//EN"
            "http://www.jboss.org/j2ee/dtd/jboss.dtd">
<jboss>
  <enterprise-beans>
    <session>
      <ejb-name>SessionEJB</ejb-name>
      <jndi-name>ejb/SessionEJB</jndi-name>
    </session>
  </enterprise-beans>
</jboss>
```

Sia *jboss.xml* che *ejb-jar.xml* devono stare in una cartella chiamata */META-INF.* Non resta che salvarli in tale posizione e creare il file JAR (che chiamiamo *ioprogrammo.jar*) con il comando:

```
%JAVA_HOME%\bin\jar -cvf ../ioprogrammo.jar it/*
   META-INF/jboss.xml META-INF/ejb-jar.xml
```

Riassumendo, la struttura del JAR creato è quella mostrata in **Figura 2**.

RICHIAMARE I METODI DELL'EJB

Per poter utilizzare il metodo messo a disposizione dall'EJB di esempio, è necessario creare una web-app che vi acceda e visualizzi all'utente il risultato con una qualche forma di interfaccia. È possibile realizzare una Servlet oppure una pagina JSP. In entrambi i casi è necessario reperire un riferimento all'EJB attraverso una ricerca via *JNDI;* tale funzionalità avviene secondo un nome, ovvero il nome con cui l'EJB viene registrato in *JNDI:* per esempio può essere *"ejb/SessionEJB".* Questo valore è contenuto nel descrittore *jboss.xml* che si è appena scritto. Ecco un esempio di pagina JSP chiamata *test.jsp* che crea un riferimento all'EJB, lo crea attraverso l'invocazione del metodo create della *Home* e infine invoca il metodo e ne visualizza il risultato:

```
<%@page import="javax.servlet.*" %>
<%@page import="javax.servlet.http.*" %>
<%@page import="java.io.*" %>
<%@page import="javax.naming.*" %>
<%@page import="javax.rmi.PortableRemoteObject" %>
<%@page import="it.ioprogrammo.ejbTest.*" %>
<%
 SessionEJBHome testSessionBean=null;
try {
  InitialContext ctx = new InitialContext();
  Object objref = ctx.lookup("ejb/SessionEJB");
  testSessionBean = (SessionEJBHome)
                    PortableRemoteObject.narrow(objref,
                                    SessionEJBHome.class);
} catch (Exception NamingException) {
  NamingException.printStackTrace(); }
try{
  SessionEJB beanRemote;
  beanRemote = testSessionBean.create();
%>
```

Fig. 2: **La struttura del file JAR**

COSA C'È SOTTO LA PENTOLA?

L'architettura di JBoss è divisa in strati, ciascuno dei quali gestisce funzionalità ben specifiche e ha responsabilità chiare e ben definite:

MICROKERNEL LAYER
È il "cuore" del server e utilizza la tecnologia JMX per fornire funzionalità di deploy delle applicazioni e caratteristiche per la gestione del ciclo di vita di una funzione (comprese funzionalità avanzate per il class-loading).

SERVICES LAYER
Appena sopra lo strato precedente, si situano i diversi servizi base di JBoss AS (quali gestione delle transazioni, servizi e-mail, gestione della sicurezza, pooling di oggetti e così via). Le tante configurazioni disponibili, utilizzate in fase di avvio di JBoss, si riferiscono a diversi tipi di servizi abilitati. Tali configurazioni agiscono proprio su questo livello: esse permettono di calibrare opportunamente solo i servizi necessari. È anche possibile realizzare servizi aggiuntivi, che si collocano in questo strato; essi vanno resi disponibili come pacchetti di tipo SAR.

SAR: SERVICE ARCHIVE
Sono dei formati proprietari di JBoss per il deploy di nuovi servizi. Essi possono essere installati anche durante il funzionamento di JBoss, rendendo l'ambiente particolarmente flessibile anche in ambienti di produzione, dov'è sconsigliato fermare un qualsivoglia server anche solo per aggiornarne l'architettura e le funzionalità

ASPECT LAYER
Si basa su un nuovo paradigma di programmazione, l'Aspect-Oriented Programming (AOP). Questo strato è accessibile anche da quello successivo (applicativo) permettendo l'uso dei suoi costrutti e della programmazione "tag-driven" anche nelle applicazioni utente;

APPLICATION LAYER
L'ultimo strato è quello applicativo ovvero dove risiedono le applicazioni sviluppate. Tali applicazioni si possono basare su funzionalità esposte dagli strati inferiori e realizzarne di nuove.

```
<center>
<h1>IoProgrammo - esempio EJB su JBoss 4.0.1</h1>
<p> L'ultimo numero della rivista è il N.<b><%=
        beanRemote.getUltimaRivista() %></b></p>
<p align="right">
<i>Ivan Venuti</i><br />
<a href="mailto:ivanvenuti@yahoo.it">
                        ivanvenuti@yahoo.it</a>
</p>
</center>
<%
  //testSessionBean.remove();
}catch(Exception CreateException){
  CreateException.printStackTrace();}
%>
```

Anche la pagina JSP deve essere inserita in una web-app, quindi in un file compresso *WAR*. Per poter essere configurato correttamente il file *WAR* per JBoss deve contenere due descrittori: *web.xml* ovvero il solito descrittore standard; non essendoci Servlet lo possiamo omettere, e il file *jboss-web.xml*; anche nel nostro caso è praticamente vuoto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss-web PUBLIC
 "-//JBoss//DTD Web Application 2.2//EN"
 "http://www.jboss.org/j2ee/dtd/jboss-web.dtd">
<jboss-web>
</jboss-web>
```

I file *jboss-web.xml* e *web.xml*, quando presente, devono essere posti sotto una cartella chiamata */WEB-INF.* Tale directory contiene anche due sotto-cartelle: *classes* e *lib*. Nella prima vanno scritti eventuali file *.class* necessari alle JSP o servlet, nella *lib* eventuali file JAR. Nel nostro esempio si deve copiare il file ioprogrammo.jar proprio nella cartella */WEB-INF/lib* e poi procedere alla creazione del file il file *ioprogrammo.war:*

```
%JAVA_HOME%\bin\jar -cvf ../ioprogrammo.war \
WEB-INF/jboss-web.xml \
WEB-INF/lib/* test.jsp
```

L'architettura del file WAR è illustrata in **Figura 3**. L'ultimo passo è quello di creare il file *EAR*. Esso si basa sia sul file JAR creato in precedenza, per creare gli EJB, che sul file WAR, per creare l'applicazione Web che accede agli EJB. Ancora una volta è necessario fornire al file EAR un descrittore, in particolare il file *META-INF/application.xml,* che nel nostro esempio contiene:

Fig. 3: **Il contenuto del file WAR**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<application>
<display-name>IoProgrammo</display-name>
<module>
  <web>
    <web-uri>ioprogrammo.war</web-uri>
    <context-root>/ioprogrammo</context-root>
  </web>
</module>
<module>
  <ejb>ioprogrammo.jar</ejb>
</module>
</application>
```

Siamo pronti per creare il file EAR:

```
%JAVA_HOME%\bin\jar -cvf ioprogrammo.ear \
META-INF\application.xml \
ioprogrammo.jar ioprogrammo.war
```

Non resta che eseguire il deploy del file *ioprogrammo.ear;* tale deploy si esegue semplicemente copiando il file nella cartella *server/nome_istanza /deploy:*

```
set JBOSS_HOME=c:\jboss-4.0.1rc1\jboss-4.0.1RC1
copy ioprogrammo.ear %JBOSS_HOME%
                          \server\default\deploy
```

Una volta eseguito il deploy, si può accedere alla pagina di test e visualizzare il risultato (**Figura 4**).

Fig. 4: **L'esempio in esecuzione: il valore 92 è reperito dall'invocazione del metodo sull'EJB**

CONCLUSIONI

JBoss riesce a coniugare quanto di meglio sia pensabile per i prodotti Open Source: un prodotto gratuito, dal codice aperto e senza vincoli d'uso e servizi professionali (a pagamento) che permettono di far dormire sonni tranquilli al più ansioso project leader che vuole la certezza che gli eventuali problemi riscontrati in produzione siano affrontati e risolti da un'azienda esperta.

Ivan Venuti

NOTA

JBoss è un esempio di come un prodotto Open Source possa divenire un vero e proprio prodotto di riferimento e mantenere, negli anni, una stabilità e una qualità del prodotto che nulla hanno da invidiare ai più blasonati Application Server commerciali. Bisogna riconoscere che JBoss nasce, e vive, in modo anomalo rispetto ad altri prodotti Open Source. Infatti la sua community non è formata solo da volontari e programmatori che prestano il loro contributo occasionalmente, ma c'è una struttura di programmatori assunti in maniera stabile da un'azienda commerciale, pagati per realizzare il prodotto. Questo implica che da qualche parte ci sia un ritorno (in termini di guadagno) e questo non è tanto nel prodotto in sé (gratuito) ma nei servizi a corredo.

Un suono sempre pulito con i filtri

Scopriamo una tecnica che ci consente di eliminare eventuali suoni sporchi o rumori da una fonte audio. Poche righe di codice Java da applicare a file sonori di qualunque tipo

Gli appassionati di vecchia musica saranno sicuramente tentati di recuperare le vecchie incisioni in vinile, magari ormai rare o introvabili, per passarli su un formato digitale indubbiamente più robusto. Il solito grande inconveniente è che, nella maggior parte dei casi, questi dischi si trovano in condizioni non proprio ottimali. Questo articolo si prefigge di fornire uno strumento per risolvere, o almeno attenuare, una delle cause di tale problema. L'aspetto che andremo ad affrontare è conosciuto in letteratura come *DSP - Digital Signal Processing* - o anche "rumore impulsivo" e lo strumento con cui cercheremo di sopprimere tale disturbo è il *filtro mediano*. Per intenderci un rumore impulsivo è quel tipo di disturbo che avviene quando, ad esempio, la puntina dei vecchi giradischi salta per un breve istante dando quella fastidiosa sensazione di interruzione del flusso musicale. La soluzione qui presentata prescinde sia dalla codifica sia dalla frequenza di campionamento utilizzate per il passaggio analogico - digitale, lasciando così al lettore la completa libertà di scegliere il sistema che reputa più opportuno per effettuare tale operazione. L'importante è avere, alla fine di questa procedura, una serie numerica *float* che rappresenti l'andamento temporale del brano. Ma veniamo al codice vero e proprio. Il cuore di questo progetto può essere essenzialmente ridotto a sole tre classi

Conoscenze richieste
Basi di Java

Software
J2SE.1.4

Impegno

Tempo di realizzazione

1. La classe astratta *Filter* che rappresenta il "concetto" di filtro
2. La classe *MedianFilter* che eredita da *Filter* ed implementa al suo interno l'effettivo algoritmo necessario affinché tale classe sia praticamente utilizzabile
3. La classe *CircolarBuffer* che verrà incapsulata all'interno di *MedianFilter* per ottimizzarne le prestazioni.

COSA È UN FILTRO DIGITALE?

Anche se gli esperti del campo potrebbero essere non proprio d'accordo con questa definizione, potremmo dire che un filtro digitale è un oggetto che prende in ingresso una serie numerica e ne restituisce un'altra in uscita tradotta secondo un proprio "meccanismo interno". Notate che la definizione è del tutto generica. Per tale motivo inizieremo definendo la nostra classe Filter come astratta. Inoltre l'unico metodo realmente indispensabile è il metodo *filter*, che restituisce un array di tipo float contenente la serie numerica trasformata. In seguito deriveremo dalla classe astratta implementando il metodo *filter*, secondo la nostra logica di riduzione del rumore. La dichiarazione sarà la seguente

```
abstract public float[] filter();
```

Presentando gli aspetti salienti di questa classe vediamo che essa possiede una coppia di array privati

```
private float[] filtered, filteredWeight;
```

In cui vengono memorizzati il segnale in uscita dal filtraggio ed i pesi ad esso associati e una coppia di array protetti

```
protected float[] inVector, weightVector;
```

COME LEGGERE L'ARTICOLO

**Questo articolo è diviso in due parti. Nello scorrere normale del testo trovate la teoria relativa alla realizzazione di un filtro per l'eliminazione del rumore. Nel box "come farlo in pratica", trovate cinque semplici passi tali che dato in input un file contenente un suono vi applicano il filtro così come viene descritto nell'articolo e producono in output un suono depurato dal rumore.
In tutto l'articolo daremo per scontato che il suono sia stato opportunamente convertito da analogico in digitale. Perciò i segnali in ingresso prima del filtro, saranno dei numeri, così come i segnali in uscita saranno ancora dei numeri.**

In cui viene memorizzato il segnale d'ingresso e i relativi pesi. Questi ultimi possono essere caricati direttamente al momento della creazione dell'oggetto *Filter* grazie a due costruttori.

```
public Filter(float[] in){
  inVector = new float[in.length];
  weightVector = new float[in.length];
  filtered = new float[inVector.length];
  filteredWeight = new float[inVector.length];
  inVector = in;
  Arrays.fill(weightVector,1);}
public Filter(float[] in, float[] weight){
  inVector = new float[in.length];
  weightVector = new float[weight.length];
  filtered = new float[inVector.length];
  filteredWeight = new float[inVector.length];
  inVector = in;
  weightVector = weight;
}
```

Per quello che concerne il significato dell'array weight vi rimando alla nota “Verificare l'affidabilità del suono”, ciò che è interessante puntualizzare è che ogni oggetto contiene già al suo interno il segnale d'ingresso per cui, se voleste cambiare input, dovreste creare un altro oggetto. In piena filosofia Java sono inoltre presenti i metodi *getFiltered()* e *getWeight()* che restituiscono rispettivamente il segnale filtrato e i pesi associati. Ciò a cui invece va prestata attenzione sono i due metodi protetti (quindi utilizzabili solo dalle classi figlie) *setfiltered()* e *setWeight()* che permettono di memorizzare il segnale filtrato e i pesi nei loro relativi array e che, come vedremo dopo, dovranno quindi essere obbligatoriamente invocati alla fine di ogni implementazione del metodo astratto affinché le elaborazioni del filtro non vadano perse!

```
protected void setFiltered(float[] filtered){
  this.filtered = filtered; }
protected void setWeight(float[] weight){
  filteredWeight = weight;
}
```

Torniamo ora al nostro problema iniziale. Abbiamo già detto che quello che vorremmo ottenere è la cancellazione del rumore causato dal "salto" della puntina. Un esempio di questo disturbo lo trovate nella parte in alto di **Figura 2.** Come si vede, oltre ad avere un brusco gradino (ad esempio in quell'istante qualche strumento ha smesso di suonare) sono presenti due picchi isolati: proprio quest'ultimi saranno i nostri nemici che vorremmo eliminare! Per chi mastica un po' di teoria dei segnali avrà già capito che un rumore del genere si concentrerà tutto in alta frequenza e quindi verrebbe spontaneo fare un semplice filtro passa-basso, ossia un filtro che non lasci

Fig. 2: ***Ingresso e uscita di un semplice filtro passa-basso***

passare tutte quelle frequenze al di sopra di una certa soglia. Purtroppo le cose non sono così semplici: ad esempio anche il timbro di un violino o l'assolo di una chitarra elettrica ha componenti ad alta frequenza. Graficamente potete osservare gli effetti di una simile elaborazione nella parte bassa di **Figura 2**. Ciò che sentireste realmente in alcuni punti sarebbe il cosiddetto effetto "wov" che, per capirci, sarebbe quell'effetto che si aveva quando le batterie dei vecchi walk-man si stavano per scaricare e la cassetta girava più lenta del dovuto.

ELIMINARE E RICOSTRUIRE: IL FILTRO MEDIANO

Siamo finalmente arrivati al cuore del progetto! Lasciatemi fare però un'ultima considerazione molto importante. Il tipo di disturbo che stiamo trattando è fortemente distruttivo; infatti quando le puntina salta (per un motivo qualsiasi), a differenza di altri generi di rumore, perdiamo completamente l'informazione di quell'istante. Come un bravo restauratore quindi il nostro filtro si dovrà occupare non solo di eliminare il disturbo, ma anche di ricostruire la parte danneggiata basandosi sui tratti limitrofi ed ancora integri del segnale stesso. L'idea che sta alla base di questo filtro è veramente semplice e può essere così sintetizzata:

1. Effettuare una finestratura di lunghezza opportuna (torneremo più avanti su questo punto) sul segnale d'ingresso.
2. Ordinare i campioni compresi nella finestratura precedente in ordine crescente o decrescente e memorizzarli in un buffer.
3. Selezionare il campione che si trova all'esatta metà del buffer (da cui il nome filtro mediano) e mandarlo in uscita del filtro stesso.
4. Far avanzare la finestra di un campione e tornare al punto 2.
5. L'algoritmo termina quando la finestra è stata fatta scorrere lungo tutto il segnale d'ingresso.

COSA VUOL DIRE ANTICAUSALE?

Un filtro completamente anticausale basa le sue elaborazioni sui campioni futuri del segnale d'ingresso. Naturalmente ciò non è possibile per applicazioni real-time.

Fig. 3: **Ingresso e uscita del filtro mediano**

In altre parole cosa abbiamo fatto? Semplice, di volta in volta scegliamo un campione che, più probabilmente, è integro. Intuitivamente potete già capire come, questo filtro, lasci passare tutti quei campioni che non sono così diversi dai propri "vicini". Un esempio grafico lo trovate in **Figura 3**.

IN PRATICA

Ora vediamo finalmente come è stato implementato questo filtro. Innanzitutto esso sarà una sottoclasse della classe astratta filter precedentemente vista

```
public class MedianFilter extends Filter {
...
```

senza ripresentare i costruttori, possiamo dire che MedianFilter ha bisogno di un parametro in più rispetto alla sua classe madre, ossia la lunghezza della finestra. Tale informazione viene memorizzata su un attributo privato intero di nome offset:

```
public MedianFilter(float[] in, int nSampleQueue){
    super(in);
    int queue = 2*((int)(nSampleQueue/2))+1;
    offset = Math.max(queue,3);
}
```

Per quanto detto in precedenza, la finestra deve obbligatoriamente avere una lunghezza dispari, in modo tale da poter scegliere il valore mediano. Quella strana formula che vedete garantisce proprio tale disparità e vedremo che risulterà molto utile nella pratica, permettendoci di disinteressarci di questo aspetto. L'ultimo passo che rimane da compiere è implementare il metodo astratto filter che rispecchierà l'algoritmo sopra. Per ragioni di sintesi, ne riportiamo e commentiamo solo i tratti principali. Troverete comunque tutto il codice nel CD allegato alla rivista. La prima cosa da fare è inizializzare tutte le strutture dati di cui abbiamo bisogno

```
public float[] filter(){
  float[] tmp = new float[inVector.length];
  float[] weight = new float[weightVector.length];
  CircolarFloatBuffer queue=new CircolarFloatBuffer(offset);
  int i,index;
  for(i = 0; i<offset;i++)
      queue.push(inVector[i]);
  ...
```

Abbiamo cioè creato due array "d'appoggio" per effettuare le nostre elaborazioni e creato un buffer circolare (la cui implementazione trovate nel CD) che verrà utilizzato come repository dati per la nostra finestra. A noi basta sapere che tale buffer è stato progettato in maniera tale da ottimizzare sia l'inserimento sia il riordinamento dei dati tramite un quick-sort. Poiché la versione qui implementata è completamente anticausale (box5) si riempe il buffer per tutta la sua lunghezza mediante il metodo push prima di effettuare qualsiasi elaborazione. A questo punto si inizia il filtraggio vero e proprio al termine del quale si fa avanzare di un campione in avanti la finestra.

```
for(i=0; i<inVector.length-offset; i++){
  tmp[i] = medianFilter(queue.getBuffer());
  ...
  queue.push(inVector[i+offset]);}
```

Logicamente la finestra non può uscire dal segnale, per cui il limite del ciclo for è dato dallo lunghezza del segnale stesso meno la lunghezza della finestra (che si presuppone comunque molto più corta). Gli ultimi campioni quindi non vengono processati, ma sono copiati così come sono. In pratica è necessario effettuare un marge tra il segnale elaborato e gli ultimi campioni rimasti fuori. Un modo efficiente è questo:

COS'È LA CONVERSIONE ANALOGICO DIGITALE?

Quando si parla di conversione analogico-digitale si intende la trasformazione di un segnale, o più astrattamente di una funzione, continuo nel tempo in uno discreto: ossia un segnale di cui si conoscono i valori solo in determinati istanti di tempo tra loro equidistanti. Tale procedura prende anche il nome di campionamento ed ogni valore è detto campione. La figura accanto ci dà un'idea di come questa procedura operi concettualmente: in alto abbiamo il segnale analogico originale in cui ogni linea verticale tratteggiata rappresenta il valore che si preleva a quell'istante. Una cosa che viene spontaneo chiedersi è: ma se tra un campione e l'altro non ho più niente non perdo informazione? La risposta è dipende. Infatti è altrettanto immediato capire che diminuendo la distanza tra due campioni (tempo di campionamento) il segnale sarà sempre più simile a quello originale. A tal proposito esiste il Teorema del Campionamento che afferma che se *Tc<1/ (2W)*, dove W è la banda occupata dal segnale analogico, il segnale analogico è perfettamente ricostruibile a partire da quello digitale. Infine anche i valori vengono discretizzati a seconda del numero di bit utilizzati.

```
...
  System.arraycopy(inVector,inVector.length-offset,
                          tmp,inVector.length-offset,offset);
...
```

Per questo motivo sarebbe opportuno lasciare in fondo al brano un paio di secondi di silenzio, in modo tale che il tratto non elaborato sia inutile dal punto di vista musicale. Alla fine è poi necessario memorizzare e restituire le elaborazioni fatte

```
...
  setFiltered(tmp);
  setWeight(weight);
  return tmp;
}
```

Fig. 3: *Ingresso e uscita del filtro mediano*

Il metodo medianFilter è infine quello che si occupa dell'ordinamento del buffer e della selezione del valore mediano

```
private float medianFilter(float[] queue){
  Arrays.sort(queue);
  int pos = (int)(offset/2);
  return queue[pos];
}
```

CONCLUSIONI

Questo strumento trova ampio interesse in letteratura ed esistono vari metodi di utilizzo, lascio a voi il piacere di sbizzarrirvi.

Andrea Galeazzi

QUATTRO SEMPLICI PASSI PER USARE IL FILTRO

Va tenuto conto prima di tutto di due parametri fondamentali tra loro correlati: la lunghezza della finestra e la frequenza di campionamento. Infatti se vogliamo eliminare un disturbo di un secondo ed abbiamo campionato il segnale a circa 44kHz (come avviene nei CD) dovremo dimensionare la finestra a circa 88000 (cioè quasi due sec.). Ricordiamo inoltre che una finestratura di lunghezza L dà luogo ad uno sfasamento massimo pari a L/2.

```
JFileChooser chooser = new JFileChooser();
  ArrayList valList = new ArrayList();
  float[] val;
  if(chooser.showOpenDialog(null) ==
                        JFileChooser.APPROVE_OPTION){
    FileReader inFile = null;
    try {
      inFile = new FileReader(
                        chooser.getSelectedFile());
    } catch (FileNotFoundException e) {
      e.printStackTrace();
        System.exit(1);
    }
```

1 **Creare una semplice interfaccia per selezionare il file dove sono contenuti i campioni e puntare tale file. La realizziamo con un Jfilechooser.**

```
BufferedReader in = new BufferedReader(inFile);
  String line = null;
  try {
    line = in.readLine();
  } catch (IOException e1) {
    while(line != null){
      valList.add(line);
        try {
          line = in.readLine();
        } catch (IOException e2) {
          ..........
          val = new float[valList.size()];
          for(int i =0; i<val.length; i++)
              val[i] = Float.parseFloat(
                        (String)valList.get(i));
```

2 **Creare l'oggetto MedianFilter con l'array creato e la lunghezza della finestra desiderata *MedianFilter f = new MedianFilter(val,88000);***

```
        float[] out = f.filter();
```

3 **Far filtrare il segnale d'ingresso e memorizzarlo su un altro array**

```
for(int i=0; i<out.length; i++)
  System.out.println(val[i] + "\t" + out[i]);
```

4 **Scrivere il risultato ottenuto su un qualsiasi stream d'uscita (in questo caso didattico è lo schermo, ma se lo avete potreste darlo ad un decoder audio)**

VERIFICARE L'AFFIDABILITÀ DEL SUONO

In realtà la conversione analogico-digitale non è un'operazione così banale come si potrebbe pensare: essa infatti è influenzata, oltre che dalla condizione della sorgente, anche da molti altri fattori. Ad esempio è molto comune nelle conversioni "fatte in casa" che si debba o voglia convertire una sorgente mono in stereo o viceversa. Come potete capire questo può dar luogo quantomeno ad un leggero sfasamento del segnale uscente. In generale esistono degli algoritmi che permettono di dare una stima dell'affidabilità di ogni campione, ossia che indica quanto quel campione sia "credibile", assegnandogli un numero da 0 a1. Questa molte volte è una informazione per la ricostruzione locale del segnale stesso. Se non si possiede tale informazione conviene assegnare a tutti i campioni un'affidabilità pari ad 1.

Protezione di dati segreti in C#

Scopriamo come poter utilizzare le classi messe a disposizione da C# e da .NET per nascondere le informazioni e i dati sensibili delle nostre applicazioni

Da sempre la crittografia rappresenta un punto cruciale nello scambio di informazioni. Antichi crittosistemi compaiono anche nella civiltà romana, quando le comunicazioni avvenivano attraverso il cosiddetto codice di cesare, una semplice trasposizione di tre lettere dell'alfabeto che garantì per molto tempo la segretezza dei messaggi. Nell'informatica moderna il concetto di crittografia e la sua applicazione non è poi così differente, per cui è necessario proteggere dati sensibili ed informazioni importanti da utenti malintenzionati che, se ne entrassero in possesso, potrebbero compromettere la stabilità del sistema. Inoltre con l'aumento dell'utilizzo di file XML, file di testo di facile lettura, per la memorizzazione immediata di dati, la protezione è diventata un elemento fondamentale. In questo articolo vedremo come rendere più sicuri i dati sensibili sfruttando le potenzialità della crittografia, e cercheremo di capire quando, come e perché applicare i diversi algoritmi a nostra disposizione, con un occhio di riguardo ai documenti XML.

INFORMAZIONI LOCALI AL SICURO

Immaginiamo di servirci di un file XML, ad esempio il file *.config* utilizzato in .NET per configurare le applicazioni, in cui memorizziamo la stringa di connessione al nostro database. Il file avrebbe, quindi, una struttura simile a questa:

```
<?xml version="1.0"?>
<configuration>
  <dataBase>
    <connectionString>Data Source=localhost;Initial
                Catalog=MyDataBase;UID=username;
                PWD=password</connectionString>
  </dataBase>
</configuration>
```

Il testo in chiaro agevola sicuramente i compiti gestionali, ma è evidentemente un punto debole del nostro sistema. Un qualsiasi utente malintenzionato potrebbe leggere il dato e usarlo per scopi diversi. Diventa quindi un'esigenza mascherare le informazioni. Ma come?

La crittografia ci fornisce le soluzioni adeguate, dobbiamo solo capire quali adottare e come utilizzarle. La scelta dipende dall'ambito di utilizzo del dato da proteggere. In poche parole dobbiamo stabilire se le informazioni riguardano l'applicazione locale oppure devono essere scambiate tra diversi utenti o computer. Nel primo caso, uno dei sistemi più sicuri ci viene offerto dalle *Data Protection API (DPAPI)*, attraverso l'utilizzo della classe *managed DataProtection*.

Nel nostro articolo svilupperemo un tool che si occuperà di generare le stringhe criptate sulla base del testo passato in input. Creiamo quindi una nuova applicazione Windows C# e aggiungiamo un riferimento all'assembly *dpapiLibrary.dll*. Nel *form1.cs* aggiungiamo un primo controllo TextBox *txtToEncrypt* ed un secondo controllo TextBox *txtEncrypted* impostando la proprietà *TextMode* su *MultiLine*. Infine aggiungiamo un controllo di tipo button *bEncrypt*. Il risultato dovrebbe essere simile a quello visualizzato in **Figura 1**.

Fig. 1 : Come si presenta il form della nostra applicazione

Il form appena creato ci consentirà di eseguire l'encrypting delle informazioni da proteggere.
Nel gestore di evento per il tasto *bEncrypt*, inseriamo il seguente codice:

```
// importiamo il namespace
using dpapiLibrary;
// istanziamo l'oggetto DataProtection
DataProtection dp = new DataProtection(
                DataProtection.Store.USE_USER_STORE);
// trasformiamo in un array di byte la stringa da codificare
byte[] toEncrypt = System.Text.Encoding.ASCII.GetBytes(
                                    this.txtToEncrypt.Text);
// eseguiamo l'encrypt
byte[] encrypted = dp.Encrypt(toEncrypt, null);
// recuperiamo in formato stringa il testo criptato
this.txtEncrypted.Text = Convert.ToBase64String(
                                             encrypted);
```

Il codice riportato crea un'istanza della classe DataProtection impostando l'utilizzo delle credenziali utente come chiave di sicurezza da applicare. Successivamente il testo contenente le informazioni da proteggere viene convertito in un array di byte e passato come primo parametro al metodo *Encrypt*, che a sua volta ci restituisce un array di byte contenente il risultato dell'encrypting. Infine riportiamo nel form la stringa rappresentante i dati criptati. A questo punto sostituiamo la stringa contenuta nel file XML con il testo criptato, ottenendo un risultato simile a quello riportato in **Figura 2**.

```
<?xml version="1.0"?>
<configuration>
    <dataBase>
        <connectionString>Data Source=localhost;
            Initial Catalog=MyDataBase;
            UID=username;PWD=password</connectionString>
    </dataBase>
</configuration>
```

Fig. 2: ***Il file XML con le informazioni criptate***

Il metodo utilizzato dalle *DPAPI* per eseguire l'encrypting consentirebbe, a tutte le applicazioni che girano sotto lo stesso utente o nella stessa macchina, di eseguire facilmente il decrypt. Per incrementare il grado di sicurezza è possibile specificare un'ulteriore valore, chiamato *entropy*, che sarà utilizzato nel processo di encrypting e nel processo di decrypting congiuntamente alla master *key* e alla random key. L'entropy è il secondo, opzionale, parametro passato ai metodi *Encrypt* e *Decrypt* della classe *DataProtection*. Per decodificare il testo protetto possiamo utilizzare un codice simile al seguente:

```
// istanziamo l'oggetto DataProtection
DataProtection dp = new DataProtection(
                DataProtection.Store.USE_USER_STORE);
// recuperiamo il testo da decriptare
string s = this.txtEncrypted.Text;
// eseguiamo il decript
byte[] b = dp.Decrypt(Convert.FromBase64String(s), null);
// recuperiamo la stringa decriptata
string result = System.Text.Encoding.ASCII .GetString(b);
```

Le chiavi vengono mantenute dal sistema, perciò non dobbiamo preoccuparci di recuperarle per applicare la decodifica. Applicare la crittografia comporta l'aggiunta di un ulteriore strato che interviene prima della scrittura sul file e dopo la lettura del file. L'applicazione di algoritmi lenti o la codifica di dati di grandi dimensioni può causare un sensibile decadimento delle prestazioni.

SCAMBIO SICURO DI DATI CRITTOGRAFATI

Il grande vantaggio offerto dalle *DPAPI* è quello di sollevarci dall'onere del mantenimento delle chiavi, evidenziando, però, un limite che in alcuni casi risulta decisivo: la portabilità. Risulta, infatti, impossibile leggere un dato codificato da un computer o un utente differente da quello che ha in realtà eseguito l'encrypting. Per proteggere dati che devono essere scambiati tra diverse applicazioni o diversi utenti, dobbiamo adottare un sistema di crittografia diverso. Il framework ci mette a disposizione diverse possibilità, tra cui distinguiamo le seguenti categorie:

- **Algoritmi a chiavi simmetriche:** utilizzano la stessa chiave per criptare e decriptare;
- **Algoritmi a chiavi asimmetriche:** utilizzano una coppia di chiavi pubblica/privata per criptare e decriptare.

Come detto, a differenza delle *DPAPI* dobbiamo preoccuparci di proteggere le chiavi, e sicuramente non conviene metterle in chiaro nel codice dato che un utente potrebbe recuperarle utilizzando un qualsiasi decompilatore. Per superare questo problema, in uno scenario reale viene spesso utilizzato un algoritmo a chiave asimmetrica per proteggere e trasmettere la chiave, concordata tra le diverse parti, per la lettura di informazioni criptate con algoritmi a chiavi simmetriche. Sicuramente vi chiederete: visto che è più sicuro, perché non utilizziamo direttamente l'algoritmo a chiave asimmetrica? Non entrando troppo nel dettaglio, possiamo affermare che questo tipo di algoritmo risulta particolarmente lento e adatto alla crittografia di piccole quantità di dati. Al contrario, gli algoritmi a chiave simmetrica sono molto più rapidi e possono essere utilizzati per crittografare anche grandi quantità di dati. A voi la scelta.

SUL WEB

Microsoft Security
http://msdn.microsoft.com/security

Windows Data Protection
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsecure/html/windataprotection-dpapi.asp

How To Create a DPAPI Library
http://msdn.microsoft.com/library/en-us/secmod/html/secmod21.asp?frame=true

Il mio blog
http://blogs.ugidotnet.org/fabioc

CRITTOGRAFIA A CHIAVE SIMMETRICA

Nella stesura del presente articolo abbiamo deciso di approfondire l'uso dell'algoritmo a chiave simmetrica anche perché dal punto di vista del programmatore le differenze tra le due tipologie sono minime. La crittografia a chiave simmetrica, o a chiave segreta, utilizza un'unica chiave condivisa per crittografare e decrittografare i dati.
Prima di scegliere l'algoritmo da applicare è doveroso fare una breve premessa. Il namespace *System.Security .Cryptography* contiene le seguenti classi, trasposizione in .NET dei relativi algoritmi a chiave simmetrica:

Algoritmo	.NET Class	Dimensioni delle chiavi	Dimensione di default
DES	*DESCryptoServiceProvider*	64 bits	64 bits
TripleDES	*TripleDESCryptoServiceProvider*	128, 193 bits	192 bits
RC2	*RC2CryptoServiceProvider*	40-128 bits	128 bits
Rijndael	*RijndaelManager*	128, 192, 256 bits	256 bits

Tabella 1: Le classi contenute nel namespace system.security.cryptography

Tutte le classi elencate derivano dalla classe *SymmetricAlgorithm* che definisce quattro principali proprietà:

- **Key:** la chiave utilizzata per le operazioni di crittografia;
- **Mode:** definisce il tipo di operazione da eseguire durante la cifratura. Attualmente tutte gli algoritmi utilizzano le modalità *ECB*, nessuna concatenazione tra blocchi, e *CBC*, cifratura a blocchi concatenati;
- **IV:** vettore di inizializzazione utilizzato per la modalità *CBC* di cifratura a blocchi concatenati;
- **Padding:** stabilisce la modalità di elaborazione dell'ultimo blocco nel caso in cui sia più piccolo di quanto stabilito nella proprietà *BlockSize;*

Occorre soffermarsi un attimo sul significato delle proprietà *Mode* e *IV.* Impostando la proprietà mode su *CBC,* il valore di default, l'algoritmo esegue la crittografia del testo procedendo a blocchi definiti. Bisogna fare attenzione, però, che utilizzando un'unica chiave, blocchi uguali nel testo non crittografato produrranno lo stesso output nel testo crittografato, consentendo ad un hacker, in possesso anche di poche informazioni, di conoscere la struttura ed eventualmente risalire alla chiave. Utilizzando un vettore di inizializzazione *IV,* invece, è possibile di cifrare in modo diverso ogni blocco di testo inserendo nella crittografia del blocco successivo le infor-

COME REALIZZARE IL PROGETTO

CREIAMO IL PROGETTO

1 **Con Visual Studio .NET creiamo un nuovo progetto Windows C# e lo chiamiamo "EncryptionTool".**

AGGIUNGIAMO I RIFERIMENTI

2 **Aggiungiamo un riferimento all'assembly dpapiLibrary.dll per sfruttare le caratteristiche delle DPAPI.**

PREPARIAMO IL FORM

3 **Rinominiamo il file form1.cs in mainForm.cs ed aggiungiamo due controlli textbox più due controlli button.**

INSERIAMO IL CODICE PER L'ENCRYPT

4 **Con doppio click sul controllo bEncrypt aggiungiamo il codice per la codifica.**

INSERIAMO IL CODICE PER LA VERIFICA

5 **Il doppio click sul controllo bVerify ci permette di aggiungere il codice per la decodifica.**

ESEGUIAMO L'APPLICAZIONE

6 **Inseriamo un testo e clicchiamo sul tasto Encrypt. Clicchiamo sul tasto Verify per verificare l'autenticità del dato.**

mazioni tratte dal blocco precedente. Per la crittografia del primo blocco viene utilizzato, per l'appunto, il vettore di inizializzazione. Dopo aver capito come opera l'algoritmo, cerchiamo di applicarlo prendendo in esame il file XML utilizzato che ora deve essere condiviso tra diversi computer. Ipotizziamo di aver valutato attentamente le varie possibilità e deciso per l'utilizzo dell'algoritmo a chiave simmetrica *Rijndael,* che ci garantisce il giusto equilibrio di sicurezza e velocità. Estendiamo le funzionalità del nostro *EncryptionTool* aggiungendo, in una nuova scheda, una textbox per la stringa da criptare, una textbox per visualizzare la chiave generata, una textbox per visualizzare il vettore di inizializzazione generato e una textbox per visualizzare il risultato. Otteniamo così un form simile a quello riportato in **Figura 3**.

Fig. 3: **Estendiamo le funzionalità dell'EncryptionTool**

Nell'evento *click* del tasto *encrypt* aggiungiamo il seguente codice:

```
// trasformo in un array di byte il testo da criptare
byte[] b = Encoding.UTF8.GetBytes(
                                    txtToEncrypt2.Text);
// istanzio la classe dell'algoritmo e creo l'encryptor
RijndaelManaged rm = new RijndaelManaged();
ICryptoTransform ct = rm.CreateEncryptor();
// memorizzo la chiave e il vettore di inizializzazione
                                          nelle textbox
txtKey.Text = Convert.ToBase64String(rm.Key);
txtIV.Text = Convert.ToBase64String(rm.IV);
// eseguo l'encrypt utilizzando l'oggetto CryptoStream
MemoryStream ms = new MemoryStream();
CryptoStream cs = new CryptoStream(ms, ct,
                              CryptoStreamMode.Write);
cs.Write(b, 0, b.Length);
cs.FlushFinalBlock();
cs.Close();
// visualizzo il risultato
txtEncrypted2.Text = Convert.ToBase64String(
                                     ms.ToArray());
```

Come è facile intuire, un'eventuale modifica per consentire l'elaborazione utilizzando un diverso algoritmo comporterebbe solo piccoli ritocchi. Il codice si avvale della classe *MemoryStream* nella quale l'oggetto *CryptoStream* scrive il risultato della codifica utilizzando l'algoritmo *Rijndael* passato al costruttore tramite l'interfaccia *ICryptoTransform.*
Il listato utilizza e memorizza le chiavi generate dinamicamente nella creazione dell'oggetto rm, ma ovviamente è possibile, come vedremo, utilizzare le proprie chiavi. Infine, per assicurare che tutti i blocchi sono stati scritti nel *MemoryStream* viene chiamato il metodo *FlushFinalBlock* e chiuso l'oggetto *CryptoStream.* A questo punto convertiamo l'array di byte in stringa per visualizzare il risultato nel form. Per decrittografare il testo è sufficiente apportare solo poche modifiche al precedente listato:

```
// trasformo in un array di byte il testo da criptare
byte[] b = Convert.FromBase64String(
                                  txtEncrypted2.Text);
// istanzio la classe dell'algoritmo e creo l'encryptor
RijndaelManaged rm = new RijndaelManaged();
// recupero la chiave ed il vettore di inizializzazione dal form
rm.Key = Convert.FromBase64String(txtKey.Text);
rm.IV = Convert.FromBase64String(txtIV.Text);
// creo l'interfaccia per il decryptor
ICryptoTransform ct = rm.CreateDecryptor();
// eseguo il decrypt utilizzando l'oggetto CryptoStream
MemoryStream ms = new MemoryStream();
CryptoStream cs = new CryptoStream(
                       ms, ct, CryptoStreamMode.Write);
cs.Write(b, 0, b.Length);
cs.FlushFinalBlock();
cs.Close();
// visualizzo il risultato
MessageBox.Show("Testo originale:" + (char)13 +
                  Encoding.UTF8.GetString(ms.ToArray()));
```

CONCLUSIONI

Nella progettazione di un sistema di crittografia, è prestare particolare attenzione alla protezione delle chiavi utilizzate, più della segretezza dell'algoritmo. Cito, in poche parole, il principio di *Kerckhoffs* secondo il quale *"la sicurezza di un crittosistema dipende esclusivamente dalla segretezza della chiave e non dalla segretezza dell'algoritmo usato".*
Spesso, nella crittografia moderna, questo principio viene trascurato e l'attenzione viene erroneamente spostata sulla protezione dell'algoritmo. Teoricamente, non potremo che essere d'accordo, anche se la sicurezza non è mai troppa. Estendendo questo concetto possiamo anche dedurre che è inutile quanto gravoso creare dei propri crittosistemi. L'*EncryptionTool* presentato in questo articolo può essere sicuramente esteso con alcuni ritocchi. Sono, come sempre, a disposizione per chiarimenti, potete contattarmi attraverso il forum della rivista.
Buon lavoro.

Fabio Cozzolino

- **WRITING SECURE CODE II Edition** *Michael Howard* *David LeBlanc* (Microsoft Press) ISBN 0-7356-1722-8 2003
- **PRACTICAL CRYPTOGRAPHY** *Niels Ferguson* *Bruce Shneier* (Wiley Publishing) ISBN 0-471-22357-3 2003
- **.NET FRAMEWORK SECURITY** *Brian A. LaMacchia* *Sebastian Lange* *Matthew Lyons* *Rudi Martin* *Kevin T. Price* (Addison Wesley) ISBN 0-672-32184-X 2002

**Ogni volta che viene creata un'istanza di una classe per la crittografia simmetrica vengono creati una chiave ed un vettore di inizializzazione (*IV*).
È anche possibile generarne di nuovi utilizzando i metodi *GenerateKey* e *GenerateIV*.
È consigliabile utilizzare una chiave ed un *IV* diverso per ogni trasmissione.**

Un lucchetto per i nostri programmi

Un cellulare Bluetooth possiede un numero che lo identifica univocamente. E se usassimo questa caratteristica come chiave d'accesso per accedere ad applicazioni protette

Utilizza questo spazio per le tue annotazioni

È ormai prassi comune accedere a certe applicazioni immettendo Username e Password. Pensate all'accesso all'area riservata di un sito, o anche semplicemente all'accesso a una cartella condivisa. In moltissimi casi è poi diffusa l'autenticazione tramite SmartCard. Pensate all'autenticazione per utilizzare la postazione di uno sportello bancario, oppure in un ufficio. In questo articolo sfrutteremo una terza possibilità, ovvero quella di sfruttare l'identificativo univoco che caratterizza una periferica bluetooth per effettuare un'identificazione automatica con una periferica abilitata. Sappiate che il vostro cellulare, così come ogni periferica bluetooth, è dotato di un identificativo univoco. Va da se che se la macchina protetta da login e password si accorge che nei dintorni c'è un cellulare con il vostro identificativo, potrebbe sbloccarvi l'accesso senza che voi dobbiate inserire fisicamente il vostro identificativo. Ovviamente niente deve essere aggiunto al vostro cellulare, farà tutto il software presente sulla macchina. Controllerà se nelle vicinanze c'è un qualche dispositivo contenente un identificativo valido e se le trova vi concederà i permessi di accesso. Ora che abbiamo chiaro il funzionamento di questa autenticazione passiamo alla costruzione dei vari moduli che ci permetteranno di creare applicazioni sicure. Per l'implementazione sfrutteremo il package avetanaBluetooth.

IMPLEMENTAZIONE DEL MODULO BASE

Il primo punto da avere ben chiaro è che il computer che utilizziamo per i nostri scopi deve essere equipaggiato con una chiavetta bluetooth o almeno possedere una schedina bluetooth. Ovviamente questo è necessario per scambiare i dati con il cellulare. Il modulo base del nostro programma sarà quello che, interfacciandosi alla chiavetta Bluetooth con il package avetanaBluetooth, farà il discovery delle device presenti. Andrà cioè alla caccia di periferiche bluetooth posizionate nei pressi del nostro computer. La JSR 82, della quale avetanaBluetooth è

COME INIZIARE

Esistono diverse implementazioni per utilizzare la connessione Bluetooth da un computer, una di queste è avetanaBluetooth. Questo prodotto è praticamente una vera e propria implementazione della JSR 82 definita dalla SUN, JSR che spiega cosa e come deve essere strutturata la comunicazione tramite Bluetooth in Java. Questo package è gratuitamente scaricabile dal sito http://www. avetana-gmbh.de /avetana-gmbh/produkte/jsr82 .eng.xml in versione dimostrativa per sistemi Windows e Mac e gratuitamente per Linux. Le API di avetana sono una delle migliori implementazioni utilizzabili per quanto riguarda la JSR 82. È importante soprattutto ricordare la semplicità con cui sono scaricabili e configurabili nel sistema di sviluppo. Altre implementazioni sono a pagamento e non permettono allo sviluppatore di poter testare in anticipo con molta libertà le applicazioni. Questa implementazione prevede il pagamento soltanto per essere utilizzata su sistemi Windows. In fase di registrazione sul sito internet devono essere inseriti gli indirizzi Bluetooth delle periferiche (chiavette) con cui il package dovrà comunicare. Successivamente viene comunicato via email un link da cui scaricare il package, che evidentemente viene creato dinamicamente in base alle informazioni da noi fornite. La versione che scarichiamo in questo modo è utilizzabile gratuitamente per 14 giorni, poi dobbiamo procedere all'acquisto oppure sottoscrivere un altro periodo di prova.

un'implementazione, specifica che per effettuare la ricerca di device deve essere implementata l'interfaccia *DiscoveryListener*. Questa interfaccia definisce quattro diversi metodi che sono praticamente delle callback per il nostro programma, chiamate in maniera asincrona quando viene completato un determinato task. Nella documentazione del package *javax.bluetooth* troviamo le signature e la spiegazione precisa di quello che succede quando questi metodi vengono richiamati. Per lo sviluppo del nostro modulo vengono presi in considerazione, quindi realmente implementati, soltanto due metodi: *inquiryCompleted()* , che viene richiamato alla fine della ricerca delle device, e *deviceDiscovered()*, che viene lanciato per ogni device trovata nel raggio d'azione della nostra chiave Bluetooth. Iniziamo quindi con l'import dei package che riguardano la nostra applicazione

```
import javax.bluetooth.*;
import de.avetana.bluetooth.*;
import de.avetana.bluetooth.sdp.*;
```

Manterremo una lista delle periferiche vicine, all'interno del vettore *trovati* definito come segue

```
public class BTObserver implements DiscoveryListener{
    private Vector trovati;
```

Ovviamente il nostro scopo è riempire questo vettore con i numeri identificativi delle periferiche bluetooth vicine. Definiamo i due metodi dell'interfaccia *DiscoveryListener* che ci servono per implementare l'interfaccia. Ovvero *inquiryCompleted( )* e *deviceDiscovered()*

```
public synchronized void inquiryCompleted(int discType)
{
        this.notify();
}
```

Per quanto riguarda il metodo *inquiryCompleted()* notificheremo al nostro programma la fine della ricerca, visto che questa avviene in maniera asincrona e non controllabile se non attraverso questi metodi di callback. Prevediamo quindi di far bloccare il nostro programma fino a quando non viene ricevuto la chiamata *notify()*. L'altro metodo, *deviceDiscovered()*, sarà invece utilizzato per riempire un vettore con tutte le informazioni che ci interessano

```
public synchronized void deviceDiscovered(
                RemoteDevice btDevice, DeviceClass cod)
{
    try{
        System.out.println("Trovata device : "+
                        btDevice.getBluetoothAddress());
        System.out.println("Nome device:
                        "+btDevice.getFriendlyName(false));
        trovati.add(btDevice.getBluetoothAddress());
        trovati.add(btDevice.getFriendlyName(false));
    }
    catch(Exception e)
    {
        System.out.println(e.toString());
    }
}
```

Definiamo ora un metodo *iniziaRicerca()* che utilizza la classe *DiscoveryAgent*, per trovare le periferiche vicine. Quando una periferica viene scovata, automaticamente si scatena un evento che richiama il metodo *deviceDiscovered* che scrive qualche messaggio e poi aggiunge la periferica al vettore dei trovati.

```
public void iniziaRicerca() {
    try {
        LocalDevice ld = LocalDevice.getLocalDevice();
        DiscoveryAgent da = ld.getDiscoveryAgent();
        da.startInquiry(DiscoveryAgent.GIAC, this);
        synchronized(this){
            this.wait();}
        System.out.println("Ricerca device completata!!!\n");
    }
    catch(Exception e) {
        System.out.println(e.toString()); }
}
```

In questo modo abbiamo una piccola classe che fa la ricerca delle device Bluetooth, le memorizza per indirizzo e nome e poi esce. Ora dobbiamo scrivere la parte di programma che permette di memorizzare le device fidate.

INIZIALIZZAZIONE

Ovviamente deve esistere un "deposito" all'interno del quale un utente avrà inserito il numero che caratterizza una periferica abilitata all'autenticazione. Affinchè un utente possa accedere alla nostra applicazione, il numero che identifica la sua periferica bluetooth deve corrispondere a uno contenuto in un elenco di periferiche fidate. In questo articolo immeteremo manualmente gli identificativi, ovviamente potete scegliere nelle vostre applicazioni, altri metodi, come un database o qualunque altra cosa. Questa fase può essere riepilogata come segue:

```
System.out.println("Richiesta device fidate");
int num=0;
String text=null;
Vector device=new Vector();
while(true){
    text=new JOptionPane().showInputDialog(
```

CHE COSA VUOL DIRE JSR 82?

Le JSR ovvero Java Specification Request sono un insieme di specifiche che definiscono uno standard. Corrispondono un po a quelle che per il mondo internet sono le RFC. Un modulo Java è aderente alle specifiche JSR se rispetta le regole che esse richiedono. Per quanto riguarda le API per BlueTooth, le specifiche vengono definite dalla JSR numero 82. AvetanaBlueTooth è conforme a queste specifiche, perciò garantisce una piena compatibilità con tutte le applicazioni aderenti allo stesso standard.

AvetanaBluetooth è basato sui più usati stack Bluetooth. Su Windows richiede l'installazione dei driver Widcomm 1.4.2.10, su MacOSX utilizza l'Apple's System Stack e su Linux utilizza BlueZ. Vista la diversità di implementazioni di stack su cui si appoggia sono diverse anche le opzioni possibili per i diversi sistemi operativi. Su Linux è impossibile la criptazione e l'autenticazione, su Mac la criptazione di connessioni esistenti e su Windows la lettura della device locale e la ricerca di servizi che non fanno parte del gruppo standard.
http://www.avetana-gmbh.de/avetana-gmbh /produkte/Readme.xml

```
                    "NUMERO DI DEVICE FIDATE:");
  if (text!=null) break; }
try {
  num=Integer.parseInt(text); }
catch (NumberFormatException e) {
  JOptionPane.showMessageDialog(null, "Non hai
            inserito un numero valido!!", "Errore",
                    JOptionPane.ERROR_MESSAGE);
  System.exit(0); }
for (int i=1;i<=num;i++) {
  text=new JOptionPane().showInputDialog("DEVICE
                    FIDATA NUMERO "+i+":");
  System.out.println("DEVICE FIDATA NUMERO
                            "+i+":"+text);
  device.add(text); }
System.out.println("Richiesta device fidate completata");
```

Abbiamo quindi creato un deposito di identificativi fidati per la nostra applicazione. Per poterli recuperare successivamente dobbiamo salvarli in qualche maniera. Per il momento scriviamo tutto su filesystem

```
try {
  PrintStream ps;
  FileOutputStream fos=new
                    FileOutputStream("fidati.log");
  ps=new PrintStream(fos);
  for (int i=1;i<=num;i++) {
    ps.println(device.elementAt(i-1));}
  ps.close();
  fos.close(); }
catch (Exception e) {
System.out.println(e.toString());
}
```

UTILIZZO DEL MODULO

A questo punto abbiamo definito come trovare le device e come inizializzare il nostro modulo con delle device fidate. Non rimane che utilizzare questo modulo all'interno dei nostri programmi. Nella classe BTObserver, che effettua il discovery delle device inseriremo un metodo che prima richiama iniziaRicerca() che esegue il discovery delle periferiche, poi apre il file dove abbiamo salvato gli id delle periferiche fidate e poi restituisce true o false

```
public boolean isPossible()
{
  //Facciamo partire la ricerca come abbiamo prima
  //spiegato. Questa chiamata bloccherà l'esecuzione
  //del programma fino a quando la ricerca non è
  //completata. Alla fine ci troveremo in un vettore tutte
  //le device visibili in questo momento
  Vector fidati=new Vector();
  iniziaRicerca();
    try
  {   DataInputStream dis;
        FileInputStream fis=new FileInputStream(
                                    "fidati.log");
        dis=new DataInputStream(fis);
        String linea;
        while((linea=dis.readLine())!=null)
        {  fidati.add(linea);
         }
        for (int i=0;i<fidati.size();i++)
        {   String elem=(String)fidati.elementAt(i);
            if (trovati.contains(elem))
            return true;
         }
         return false; }
```

L'APPLICAZIONE IN 6 SEMPLICI PASSI

AVVIO RICERCA

```
LocalDevice ld =
                LocalDevice.getLocalDevice();
DiscoveryAgent da =
ld.getDiscoveryAgent();
da.startInquiry(DiscoveryAgent.GIAC, this);
synchronized(this){
  this.wait();
}
```

1 **Nel nostro programma possiamo istanziare un DiscoveryAgent, la classe che ci permette di ricercare device e servizi. Fino a quando la ricerca non è completata rimaniamo in attesa.**

INTERFACCIA DISCOVERYLISTENER

```
public synchronized void
            inquiryCompleted(int discType) {
  this.notify();}
public synchronized void
 deviceDiscovered(RemoteDevice btDevice,
                      DeviceClass cod){
  try{
    System.out.println("Trovata device : "+
          btDevice.getBluetoothAddress());
    System.out.println("Nome device:
    "+btDevice.getFriendlyName(false));}
  catch(Exception e) {}
}
```

2 **Per avere informazioni riguardanti le device Bluetooth che sono trovate durante lo scan dobbiamo implementare l'interfaccia DiscoveryListener che con diverse callback permette di avere informazioni sullo stato della ricerca.**

INIZIALIZZAZIONE DEVICE AUTORIZZATE

```
for (int i=1;i<=num;i++) {
  text=new JOptionPane().
        showInputDialog("DEVICE FIDATA
                  NUMERO "+i+":");
  System.out.println("DEVICE FIDATA
              NUMERO "+i+":"+text);
  device.add(text);
}
System.out.println("Richiesta device fidate
                        completata");
```

3 **Prima di utilizzare il nostro modulo dobbiamo inizializzarlo, richiedendo una lista di device autorizzate ad avviare l'applicazione. Per questo esempio useremo una semplice dialog box.**

```
      catch (Exception e)
    { System.out.println(e.toString());
      return false; }
}
```

Con questo metodo controlliamo semplicemente la presenza di device conosciute come sicure per l'avvio del nostro programma. Quindi per utilizzare questa classe in un'altra applicazione dobbiamo semplicemente istanziarla e richiamare il metodo *isPossible()*

```
BTObserver bto = new BTObserver();
boolean start=bto.isPossible();
if (start)
   avviaProgramma();
else
   esciProgramma();
```

Nel caso in cui non ci siano delle device fidate nelle vicinanze il nostro programma esce, richiamando il metodo *esciProgramma()*, nel quale possiamo definire un messaggio d'errore per l'utente

```
public void esciProgramma(){
   JOptionPane.showMessageDialog(null, "Nessuna
   device autorizzata è stata trovata nelle vicinanze!!",
               "Errore", JOptionPane.ERROR_MESSAGE);
   System.exit(0);
}
```

ULTERIORE PASSO DI SICUREZZA

Ora siamo sicuri che la nostra applicazione verrà avviata solo in determinate circostanze. Rimane però un piccolo problema, la presenza sul nostro computer di un file di testo con elencate le device Bluetooth che il programma considera fidate. Dobbiamo chiaramente escogitare un modo per alzare il livello di sicurezza anche per quanto riguarda questo file. Per non lasciare questo file in chiaro nel nostro computer possiamo appoggiarci a JCE (Java Cryptography Extension). Usando un algoritmo di crittografia come 3DES o RSA possiamo lasciare questo file sul computer e non avere paura se qualcuno lo legge (visto che sarebbe ben poco quello che potrebbe capirne). Oppure potremmo utilizzare SHA-1 per lasciare un'impronta digitale del file di testo, per essere sicuri che non sia stato modificato. Tutte queste sono scenari ammissibili pensando ad un programma che non rimane sempre in esecuzione. Se invece parliamo di un server o comunque di un'applicazione che necessita di dover rimanere sempre avviata possiamo pensare a immagazzinare i dati direttamente nell'applicazione, senza avere problemi di memorizzazione e crittografia.

Federico Paparoni

COME RICAVARE L'ID DI UNA PERIFERICA BLUETOOTH

Abbiamo diverse modi per trovare l'indirizzo Bluetooth da inserire nella nostra applicazione. Se disponiamo di un cellulare Symbian possiamo semplicemente digitare il codice *#2820# e l'indirizzo ci viene visualizzato a schermo. Per gli altri cellulari non c'è qualcosa di standard quindi dobbiamo trovare l'indirizzo dal computer con una chiavetta Bluetooth. Dopo che abbiamo installato la chiavetta abbiamo la cartella "Risorse di rete Bluetooth" sotto "Risorse del computer". Cerchiamo tutte le device Bluetooth visibili e clicchiamo col tasto destro su Proprietà: ci viene mostrata una schermata. Per utilizzare questo indirizzo Bluetooth nella nostra applicazione dobbiamo togliere i due punti che si trovano all'interno dell'indirizzo e inserirlo tranquillamente nel programma di inizializzazione delle periferiche.

MEMORIZZAZIONE DEVICE

```
PrintStream ps;
FileOutputStream fos=new
FileOutputStream("fidati.log");
ps=new PrintStream(fos);
for (int i=1;i<=num;i++) {
   ps.println(device.elementAt(i-1));
}
ps.close();
fos.close();
```

4 **Per poter far sapere al nostro modulo quali sono le device autorizzate le scriviamo su filesystem tramite una semplice dialog box (sistema non troppo sicuro e che quindi può essere migliorato).**

EFFETTUARE IL CONTROLLO

```
DataInputStream dis;
FileInputStream fis=new
FileInputStream("fidati.log");
dis=new DataInputStream(fis);
String linea;
while((linea=dis.readLine())!=null) {
   fidati.add(linea); }
for (int i=0;i<fidati.size();i++) {
   String elem=(String)fidati.elementAt(i);
   if (trovati.contains(elem))
                  return true;
}
```

5 **Effettuiamo la ricerca e memorizziamo tutte le device trovate in un Vector. Non rimane che controllare queste device con quelle autorizzate e concedere o meno all'utente l'accesso all'applicazione.**

UTILIZZO DEL MODULO

```
BTObserver bto = new BTObserver();
boolean start=bto.isPossible();
if (start)
   avviaProgramma();
else
   esciProgramma();
```

6 **Una volta che il modulo è stato inizializzato, possiamo semplicemente istanziarlo da un altro programma e richiamare il metodo *isPossible()* per sapere se sono verificate le condizioni di sicurezza necessarie.**

La programmazione orientata agli oggetti

Il termine orientato agli oggetti, negli anni passati era considerato un argomento delicato e difficile da comprendere, ma oggi è impossibile non imparare la OOP

Utilizza questo spazio per le tue annotazioni

Il concetto fondamentale della programmazione ad oggetti è la suddivisione del programma in parti isolate, e indipendenti dalle altre, denominate oggetti. Gli oggetti sono concettualmente molto simili ai blocchetti dei giochi di costruzione, se programmiamo per bene gli oggetti possiamo disporli per costruire qualcosa di più grande della sola somma delle parti. Per accedere ai dati di un oggetto si deve semplicemente dare un comando all'oggetto che li contiene. Non è mai possibile accedere direttamente ai dati ma soltanto ai comandi che consentono l'accesso ad essi. Per modificare il modo in cui il programma accede alle informazioni di un oggetto, si devono quindi, cambiare soltanto i comandi all'interno dell'oggetto stesso.

DEFINIZIONI DI CLASSI ED OGGETTI

In letteratura si definisce oggetto: un insieme correlato di dati *(proprietà)* e funzioni *(metodi)* che agiscono su tali dati. Oggetti diversi con caratteristiche simili sono raggruppabili in una classe. Così tutte le riviste di un edicola sono oggetti della classe rivista, e tutte le persone sono oggetti della classe persona. Anche per le proprietà ed i metodi si può fare una similitudine con il mondo reale. La rivista ioProgrammo ha un titolo ed una casa editrice definiti tramite le sue proprietà. Per conoscere il mese in cui è stato pubblicato l'articolo sulla OOP, si deve fare riferimento ad una funzione o metodo dell'oggetto *ioProgrammo,* che cerca e restituisce il mese in esame. Sempre in letteratura, si afferma che un oggetto è un'istanza di una classe. È bene chiarire dal punto di vista della programmazione la differenza tra classe ed oggetto. Una classe è una parte di programma che contiene il codice necessario alla definizione di proprietà, metodi ed eventi di oggetti che saranno creati in fase di esecuzione. Un oggetto, invece, è un area di memoria creata in fase di esecuzione. L'utente non utilizzerà mai una classe, ma piuttosto utilizzerà gli oggetti creati dalle classi, ad esempio i dati di un cliente o lo stato civile di un cittadino. Il programmatore, viceversa, si troverà a scrivere il codice per definire una classe.

CARATTERISTICHE E VANTAGGI DELLA OOP

Incapsulamento è certamente la caratteristica più importante della OOP, in pratica, i dati sono proprietari dell'oggetto e sono accessibili e modificabili solo attraverso i suoi metodi. L'incapsulamento permette di creare classi facilmente riutilizzabili che non dipendano da un'altra entità esterna. Per lo stesso motivo si deve tentare di rendere la classe indipendente da procedure generiche posizionate in un altro modulo. Quando non se ne può fare a meno, è preferibile duplicare tali procedure in ogni modulo di classe.

Ereditarietà è la capacità di una classe (classe derivata o sottoclasse) di ereditare, od ottenere funzionalità di un'altra classe (la classe base o superclasse). In questo modo la classe derivata eredita automaticamente le proprietà ed i metodi della superclasse. La parola chiave utilizzata in VB .Net per creare una relazione di eredità è *Inherits*; questa istruzione deve essere sulla prima riga di codice dopo la definizione della classe. Nelle righe successive, sarà possibile scrivere il codice che definisce altre proprietà, metodi, ecc peculiari della classe derivata. I vantaggi dell'ereditarietà possono essere riassunti in:

- **Riusabilità.** Una classe può essere costruita in modo indipendente da uno specifico contesto, poiché tutte le caratteristiche generali sono scritte una sola volta
- **Portabilità.** Cambiando il contesto, è sufficiente cambiare la classe base, senza modifiche sulla parte applicativa
- **Riduzione del codice.** Non è necessario riscrivere tutti i metodi e le proprietà della classe base

ma soltanto quelli peculiari della classe derivata.

Polimorfismo significa “molte forme”. Questo termine applicato alla OOP, indica la caratteristica per cui classi diverse hanno metodi con lo stesso nome che implementano comportamenti diversi. Un esempio classico, in un’applicazione gestionale, si ottiene quando è necessario salvare i dati delle classi *Articolo* e *Cliente* nel DataBase, tutte e due le classi avranno il metodo Salva, ma naturalmente esso sarà implementato in maniera differente poiché i dati della classe cliente sono differenti dai dati della classe *Articolo.* Un altro esempio è quello di oggetti completamente diversi tra loro quali ad esempio: *TextBox, ComboBox, ListBox* che espongono alcuni metodi e proprietà uguali. Grazie a questa caratteristica, il programmatore non avrà necessità di ricordare nomi e sintassi diverse.
Overloading sotto certi aspetti è simile al polimorfismo. L’overloading, che letteralmente significa sovraccaricamento, consiste nella possibilità di definire più versioni di uno stesso metodo all’interno di una data classe, utilizzando lo stesso nome ma un numero e/o un tipo diverso di argomenti. In fase di esecuzione, sulla base degli argomenti effettivamente passati alla routine, verrà richiamata la procedura corretta.

LE PROPRIETÀ

Per definire una proprietà, il modo più semplice è quello di dichiarare una variabile pubblica nel modulo di classe. Questo metodo richiede una riga di codice per ogni proprietà. Supponendo di voler implementare la classe *film:*

```
Public Titolo As String
Public Genere As String
Public Costo As Decimal
```

La parola chiave *Public*, rende visibile la variabile da qualsiasi parte del codice che utilizza l’oggetto.
Anche se questo è il metodo più semplice per definire le proprietà in una classe, non è il metodo raccomandato poiché verrebbe a cadere il requisito dell’incapsulamento che abbiamo già visto essere un requisito fondamentale della OOP. Per evitare i rischi di variabili *Public,* le variabili che definiscono le proprietà della classe si devono dichiarare *Private.*

```
Private mTitolo As String
Private mGenere As String
Private mCosto As Decimal
```

Per convenzione e bene far precedere da un prefisso le variabili private, che dovranno definire le proprietà della classe. Nel nostro esempio utilizziamo il prefisso *m*, per distinguere le variabili private dalle proprietà che la classe renderà disponibile. Soltanto le procedure all’interno del modulo di classe potranno quindi modificare il valore di queste variabili. È possibile utilizzare anche la classica istruzione *Dim,* per dichiarare variabili visibili soltanto all’interno della classe.

LE PROCEDURE PROPERTY

A questo punto la domanda diventa: se le proprietà devono essere dichiarate *Private* e pertanto non possono essere modificate dall’esterno, a cosa servono? Per questo ci viene in aiuto il metodo *Property.* All'interno di un blocco *Property ... End Property* si può utilizzare:

- il costrutto **Get. ..End Get** che permette di

Nelle precedenti versioni di VB l’uso della parola chiave Nothing era molto importante ed obbligata. In VB.Net, con l’implementazione del meccanismo di Garbage Collection, non viene più garantito il rilascio immediato degli oggetti. La documentazione consiglia di assegnare a Nothing solo oggetti con durata estesa, come membri condivisi o variabili globali

CREAZIONE DI UNA CLASSE

Mettiamo subito in azione le classi creando un nuovo progetto Windows Applications dal nome Videoteca

1 **Selezioniamo dal menu *Progetto* la voce aggiungi classe. Verrà visualizzata la finestra di dialogo *Aggiungi nuovo elemento*.**

2 **Nella finestra di dialogo *Aggiungi nuovo elemento*, sarà selezionato il modello *Classe* (nella parte destra della maschera).**

3 **Nel campo *Nome* dobbiamo inserire il nome della classe. Per il nostro esempio possiamo scrivere: *film.***

4 **Clicchiamo sul pulsante *Apri*. Si aprirà la finestra di progettazione del codice di VB.NET con il cursore posto all’interno della classe.**

Una tipica classe consiste di tre parti:
• Dichiarazione delle variabili che utilizzerà soltanto la classe
• Dichiarazioni delle proprietà
• Implementazione dei metodi ossia le procedure o funzioni che gestiscono le variabili e le proprietà.

ottenere il valore di una proprietà da altre parti del programma. Il codice all'interno del costrutto dovrà preoccuparsi di restituire il valore della proprietà richiesta. Si può includere al suo interno qualsiasi altro codice (ad es. calcoli oppure conversione di dati).

- il costrutto **Set...End Set** che permette di impostare il valore di una proprietà da altre parti del programma. Anche in questo costrutto si può scrivere qualsiasi altro codice, ad esempio per la convalida e conversione dei dati o per assegnare un valore di default alla proprietà.

Nel nostro esempio, per definire la proprietà *Titolo*, dovremo scrivere:

```
Property Titolo() As String
    Get
        Return mTitolo
    End Get
    Set(ByVal Valore As String)
        mTitolo = Valore
    End Set
End Property
```

Il codice all'interno di *Set* verrà eseguito ogni volta che si assegna un valore alla proprietà *Titolo*. Il valore della proprietà viene passato come parametro alla procedura, per valorizzare la variabile interna *mTitolo*.
Se scriviamo:

```
ObjFilm.Titolo = "Gli Incredibili"
```

viene valorizzata la proprietà *Titolo* dell'oggetto *ObjFilm* pari alla stringa *"Gli Incredibili"*. La sintassi del costrutto *Get* è simile a quella di una funzione che restituisce un valore dello stesso tipo della proprietà. Se scriviamo:

```
TextBoxTitolo.Text = ObjFilm.Titolo
```

DISEGNO DELL'INTERFACCIA UTENTE DELLA NUOVA APPLICAZIONE VIDEOTECA

1 Selezioniamo la finestra *Form1* e, se non è già visualizzata, apriamo la finestra delle proprietà. Dalla finestra delle proprietà selezioniamo la proprietà *Name* e cambiamo subito il nome in: *FormVideoteca*. Selezioniamo la proprietà *Text* e modifichiamo il testo visualizzato nella barra del titolo in: *Gestione della tua Videoteca.*

2 Selezioniamo per quattro volte un controllo *TextBox* dalla casella degli strumenti (nella sezione Windows Form) e disegniamo i quattro controlli sulla form.
Possiamo posizionarli usando la griglia della form e agganciando gli estremi a punti sensibili.

3 Selezioniamo il primo *Textbox* e, dalla finestra delle proprietà, evidenziamo *Name* per cambiare il nome in: *TextBoxTitolo*. Evidenziamo la proprietà *Text* e cambiamo il testo nella stringa vuota. Selezioniamo gli altri *TextBox* e variamo la proprietà *Name* rispettivamente in: *TextBoxGenere, TextBoxCosto, TextBoxGiorni* e le proprietà *Text* a vuoto.

4 Selezioniamo per quattro volte un controllo Label dalla casella degli strumenti e disegniamo i quattro controlli sulla form in corrispondenza dei quattro *TextBox* disegnati in precedenza.

5 Selezioniamo la prima Label e, dalla finestra delle proprietà, evidenziamo la proprietà *Text* per cambiare il testo visualizzato in: *Titolo*. Selezioniamo le altre Label e variamo la proprietà Text in: *Genere, Costo, Numero Giorni.*

6 Selezioniamo un controllo *Button* dalla casella degli strumenti e disegniamolo sulla form.
Anche questo può essere posizionato in modo preciso utilizzando la griglia sulla form.

Viene chiamato il codice all'interno del costrutto *Get* che restituisce il valore della variabile privata *mTitolo*, per questo nel *TextBox* verrà visualizzata la stringa *"Gli Incredibili"*. Analogamente per le altre proprietà dovremo scrivere:

```
Property Genere() As String
    Get
        Return mGenere
    End Get
    Set(ByVal Valore As String)
        mGenere = Valore
    End Set
End Property
Property Costo() As Decimal
    Get
        Return mCosto
    End Get
    Set(ByVal Valore As Decimal)
        mCosto = Valore
    End Set
End Property
```

In relazione alla presenza dei costrutti *Get. ..End Get* e *Set...End Set*, le proprietà possono essere:

- **Proprietà a sola lettura.** Per creare una proprietà di sola lettura, è sufficiente omettere il costrutto *Set*.
- **Proprietà a sola scrittura.** Per creare una proprietà di sola scrittura, è sufficiente omettere il costrutto *Get*.
- **Proprietà accessibili in lettura e scrittura.** Per creare una proprietà accessibile in lettura e scrittura, è necessario implementare i due costrutti *Get* e *Set*.

7 **Selezioniamo il controllo Button e, dalla finestra delle proprietà, modifichiamo la proprietà Name in: ButtonMostraDati e la proprietà Text in: Mostra i Dati del Film. Nei quattro TextBox d'input, l'utente dovrà inserire i dati inerenti al film ceduto in noleggio. Quando l'utente clicca con il mouse sul bottone, sarà lanciata la procedura di evento ButtonMostraDati_Click che mostrerà un messaggio riassuntivo con i dati del film, ed il costo dovuto per la durata in giorni del noleggio.**

METODI

Finora abbiamo definito le proprietà della classe film, a questo punto possiamo scrivere le procedure o le funzioni che potranno svolgere operazioni con i dati memorizzati nelle proprietà, tali procedure o funzioni sono dette metodi. Per definire un metodo è sufficiente quindi scrivere una *Sub* oppure una *Function* all'interno di una classe. L'implementazione di un metodo sarà nascosta ad altre parti dell'applicazione. Nella classe film, del nostro esempio, si potrà scrivere il seguente metodo che calcola il costo di affitto di un film:

```
Public Function CalcolaCosto(ByVal NumeroGiorni As
                                    Integer) As Decimal
    CalcolaCosto = Costo * NumeroGiorni
End Function
```

Dove *Costo* è la proprietà della classe e *NumeroGiorni* è un valore passato al metodo. Nella scrittura del metodo si usano i nomi delle proprietà al posto dei nomi interni delle variabili, che sono comunque sempre visibili all'interno della classe, questo comportamento è sempre auspicabile poiché in questo caso viene sempre eseguito il codice all'interno della *Property*. Per definizione, un metodo deve essere dichiarato *Public* perché sia visibile all'esterno della classe. Ogni metodo definito come pubblico può essere eseguito in qualsiasi parte del codice per un determinato oggetto.
Se vogliamo scrivere, invece, una routine che viene utilizzata soltanto all'interno della classe, e che non è di alcun utilizzo all'esterno di essa, possiamo dichiarare la routine come Private.
Nella definizione di una routine è possibile usare anche la parola chiave Protected. Dichiarando una routine di una classe come Protected, essa non sarà accessibile alle istanze della classe ma sarà ereditabile da eventuali classi derivate. In pratica, Protected equivale a Private, con la differenza che il metodo è visibile anche alle classi che ereditano da quella principale.

LA VARIABILE OGGETTO

La creazione di un oggetto viene detto creazione di un'istanza, ed in VB.Net 2003 può essere trattato come una vera e propria variabile. Una volta creati gli oggetti, ognuno avrà i propri valori e potrà eseguire i propri metodi.
Tipicamente per un uso corretto di una variabile oggetto, si distinguono le seguenti fasi:

- Dichiarazione della variabile *oggetto*
- Creazione dell'*oggetto*
- Uso delle proprietà ed i metodi dell'*oggetto*
- Rilascio dei riferimenti all'*oggetto*

NOTA

NEW

Nella fase di dichiarazione è possibile utilizzare la parola chiave *New*, in questo modo la variabile oggetto sarà creata immediatamente.

```
Dim ObjFilm As New film
```

VARIABILE OGGETTO

Una variabile oggetto può essere utilizzata allo stesso modo di una variabile normale (utilizzando ad esempio le istruzioni *Dim, Private* o *Public*) e presenta le stesse caratteristiche di visibilità (scope) e di durata (*lifetime*). Così come le normali variabili, VB consente di dichiarare la visibilità delle variabili oggetto a livello di:

- **Blocco**
- **Procedura**
- **Modulo**
- **Progetto**

Quando eseguite l'applicazione si potrebbe verificare l'errore Impossibile trovare "*Sub Main*" in "*Videoteca .Form1*". In questo caso dovete selezionare la voce di menu: *Progetto-->Proprietà di Videoteca* in modo da visualizzare la *Pagina di Proprietà di Videoteca.* Nella sezione *Generale* (parte sinistra della finestra) dovete selezionare nel campo *Oggetto di avvio* (parte destra) la form *formVideoteca.*

Una variabile oggetto può essere dichiarata in qualunque parte del codice con la stessa sintassi di una normale variabile. A differenza delle variabili normali, che possono essere utilizzate non appena vengono dichiarate, per utilizzare una variabile oggetto si deve creare esplicitamente il riferimento all'oggetto stesso. Se tentiamo di accedere ad una proprietà di un oggetto, che non sia stato creato esplicitamente, il compilatore genera un'eccezione *"Riferimento a un oggetto non impostato su un'istanza di oggetto"*. Dopo aver creato l'oggetto, sarà possibile utilizzare una qualsiasi proprietà, provocando l'esecuzione delle procedure *Property* definite nella classe, ed un qualsiasi metodo provocando l'esecuzione delle relative procedure. Dopo aver utilizzato la variabile oggetto è buona norma distruggerla esplicitamente, anche se in VB.Net 2003 è implementato un meccanismo che si occupa del rilascio automatico delle risorse. Vedremo in azione i singoli passi nel codice di esempio.

Fig. 1: *L'applicazione in esecuzione*

FINESTRA DEL CODICE

```
Private Sub
ButtonMostraDati_Click(
    ByVal sender As System.Object,
      ByVal e As System.EventArgs)
     Handles ButtonMostraDati.Click
End Sub
```

1 **Per scrivere il codice è sufficiente fare doppio click sul controllo Button. Con questa operazione si aprirà la finestra del codice con il cursore posto all'interno della procedura di evento ButtonMostraDati_Click in cui scriveremo il nostro codice**

DICHIARAZIONE DELLA VARIABILE OGGETTO

```
Dim ObjFilm As film
Dim CostoTotale As Decimal
Dim Messaggio As String
```

2 **Una variabile oggetto può essere dichiarata allo stesso modo di una variabile normale con alcune caratteristiche specifiche:**

- **Possibile utilizzo della parola chiave New nella dichiarazione della variabile**
- **L'argomento di tipo classe.**

La variabile oggetto ObjFilm rappresenta un riferimento ad un oggetto della classe film. Dichiariamo inoltre le variabili CostoTotale e Messaggio.

CREAZIONE DELL'OGGETTO

```
ObjFilm = New film
```

3 **Per creare nuovi oggetti, con le caratteristiche definite nella classe corrispondente, si deve utilizzare la parola chiave New. Quando la linea di codice viene eseguita, viene creato un oggetto ObjFilm, istanza della classe film, e viene eseguito il metodo costruttore che analizzeremo nel prossimo articolo. Dopo aver creato l'oggetto si possono utilizzare le proprietà ed i metodi definiti nella classe.**

USARE LE PROPRIETÀ DELL'OGGETTO

```
ObjFilm.Titolo =
TextBoxTitolo.Text
ObjFilm.Genere =
                TextBoxGenere.Text
ObjFilm.Costo = CDec(
                 TextBoxCosto.Text)
```

4 **Per accedere alle proprietà di un oggetto si deve usare il punto (NomeOggetto.NomeProprietà). VB visualizza automaticamente, dopo il punto, un elenco da cui poter selezionare tutte le proprietà, metodi ed eventi utilizzabili.**

USARE I METODI DELL'OGGETTO

```
CostoTotale =
ObjFilm.CalcolaCosto(
       CInt(TextBoxGiorni.Text))
```

5 **Per accedere ai metodi si deve utilizzare, allo stesso modo delle proprietà, il punto dopo la variabile oggetto.**

MESSAGGIO RIASSUNTIVO

```
Messaggio = "L'importo dovuto
        per il noleggio del film:"
Messaggio = Messaggio &
                    ObjFilm.Titolo
Messaggio = Messaggio & " é di
            Euro:" & CostoTotale
MessageBox.Show(Messaggio)
```

6 **Componiamo il messaggio riassuntivo con il titolo del film ed il costo totale dovuto per il noleggio, e lo mostriamo a video utilizzando l'oggetto MessageBox**

I RIFERIMENTI ALL'OGGETTO

```
ObjFilm = Nothing
```

7 **In VB. Net non è obbligatorio distruggere implicitamente una variabile oggetto quando non deve più essere utilizzata nel codice, comunque può essere ancora utile mantenere questa abitudine.**

LE VARIABILI OGGETTO

Una variabile *oggetto* non è un area di memoria che contiene i dati dell'oggetto, ma è piuttosto un puntatore ad un area di memoria in cui sono memorizzati i dati dell'oggetto. Questo concetto deve essere sempre presente nella fase di programmazione poiché ne conseguono alcune caratteristiche peculiari molto importanti:

- Ogni variabile *oggetto*, essendo un puntatore ad un area di memoria, occuperà sempre lo stesso spazio indipendentemente dalle dimensioni dell'oggetto cui fa riferimento (quattro byte)
- Ogni volta che si pone una variabile oggetto uguale ad un'altra, non verrà duplicato nessun dato ma in realtà verrà assegnato alla nuova variabile oggetto lo stesso indirizzo in memoria della precedente.
- Se due o più variabili oggetto indicano la stessa istanza dell'oggetto, esisterà soltanto una copia delle proprietà, per questo modificando il valore di una proprietà di una qualsiasi di queste variabili, tutte le altre variabili vedranno immediatamente il nuovo valore

CONCLUSIONI

Con la programmazione ad oggetti, e con un po' d'impegno da parte del programmatore, è ormai abbastanza facile produrre un codice riusabile e facilmente mantenibile, per questo non perdetevi d'animo e continuate a seguirci nei meandri della OOP.

Luigi Buono

DataGrid avanzate con ASP.NET

La modifica dei dati di un database è operazione che spesso, richiede la scrittura di molto codice e l'inserimento di operazioni ripetitive e noiose. ASP.NET ci aiuta con la DataGrid

Utilizza questo spazio per le tue annotazioni

Nel numero precedente di questa serie abbiamo visto come creare, con pochi semplici passaggi, delle pagine ASP .NET in grado di mostrare dati prelevati da un database o da una qualsiasi fonte dati, come un file XML. Abbiamo intravisto l'utilizzo del DataGrid, rimandandone una trattazione più approfondita proprio a questo articolo.

IL DATAGRID AI NOSTRI PIEDI: LE BOUNDCOLUMN

Il DataGrid implementa di default funzionalità che consentono di modificare i dati attraverso delle comode griglie. Chi viene dallo sviluppo di applicazioni Windows, del resto, avrà già familiarità con l'argomento. In precedenza abbiamo visto come il DataGrid, unico tra i tre *Data Controls*, sia in grado di generare automaticamente le colonne in base al contenuto della fonte dati.
Ci sono casi in cui però lo stile di default non è sufficiente, perché bisogna dotare l'applicazione di un certo look o più semplicemente bisogna personalizzarne le funzionalità. In casi come questo è necessario utilizzare un particolare tipo di oggetti, chiamati *Column*, che consente di personalizzare con pochi e semplici mosse il layout della griglia. Per prima cosa bisogna impostare la proprietà *AutoGenerateColumns* del DataGrid su *false*, perché di default il suo valore è true e l'effetto ottenuto è quello appena menzionato. Fatto questo sarà necessario aggiungere le nostre colonne, in modo da visualizzare i dati. Per farlo si sfrutta un tipo di *Column* chiamato *BoundColumn*, che nel nome già indica la propria funzionalità: si tratta di un control che mostra i dati prelevati dal data source. Per ogni colonna da inserire bisognerà aggiungere un *<asp:BoundColumn />* ed il vantaggio di questo control è che in fase di editing offre la possibilità, per la colonna, di mostrare in automatico un tag *<input />* senza ulteriore intervento da parte nostra. Per personalizzare il layout di ogni colonna si potranno utilizzare queste proprietà:

- **HeaderText:** per specificare un'intestazione;
- **FooterText:** per indicare un footer;
- **HeaderStyle, FooterStyle, ItemStyle e AlternatingItemStyle:** per associare stili ad intestazione, footer, elemento ed elemento alternato;
- **DataFormatString:** per applicare una trasformazione al testo del campo, ad esempio formattare una data o un decimale.

Oltre a *BoundColumn* esistono altri tipi di colonne in grado di garantire una personalizzazione del DataGrid ed ereditano tutti dalla classe *DataGridColumn* contenuta nel namespace *Sytem.Web.UI.WebControls*, così che risultano facilmente estendibile attraverso oggetti custom:

- **ButtonColumn:** una colonna come pulsante;
- **EditCommandColumn:** contiene i link alle funzionalità di editing del DataGrid.
- **HyperLinkColumn:** offre una colonna con un link;
- **TemplateColumns:** l'utilizzo dei solito template, già visti con gli altri *Data Controls*, per personalizzare al massimo il risultato visivo.

BUTTONCOLUMN

Come detto *ButtonColumn* non fa altro che offrire un pulsante all'utente. Le proprietà più interessanti di questo controllo sono:

- **ButtonType:** il pulsante può un *LinkButton* o un *PushButton* (link o button HTML);
- **CommandName:** la funzione da invocare alla pressione;
- **DataTextField:** il nome del campo da cui sarà preso il testo visualizzato nel link;
- **DataTextFormatString:** il campo da formattare predendo il valore della proprietà precedente, sostituito a *{0};*
- **Text:** è il testo da visualizzare come descrizione del control, quando non si vuole recuperare dalla fonte dati.

Sono supportate le stesse proprietà di *BoundColumn* (che per questo non sono state riportate), visto che entrambi i control ereditano dalla stessa classe base.

HYPERLINKCOLUMN

HyperLinkColumn è una colonna che inserisce semplicemente un link. Le sue proprietà per questo motivo sono orientate proprio a questa funzionalità:

- **DataNavigateUrlField:** il campo da cui prendere il valore per il link;
- **DataNavigateUrlFormatString:** specifica il formato per il link, che è qualcosa come *link.aspx?id={0}*, dove *{0}* è il valore del campo specificato nelle proprietà precedente.
- **DataTextField e DataTextFormatString:** già trattati;
- **NavigateUrl:** un url fisso a cui puntare;
- **Target:** il target (HTML) a cui far puntare;
- **Text:** un testo fisso associato alla descrizione del control.

Un tipico esempio di utilizzo sarà

```
<asp:HyperLinkColumn runat="server"
  HeaderText="Ordine"
  DataNavigateUrlField="OrderID"
  DataNavigateUrlFormatString="ordine.aspx?id={0}"
  DataTextField="OrderID"
  DataTextFormatString="Ordine n. {0}" />
EditCommandColumn
```

Prima di lanciarci in funzionalità di modifica, dobbiamo analizzare quest'ultimo tipo di colonna perché ha un ruolo centrale nelle funzionalità di editing del DataGrid. Si tratta di un control che permette di aggiungere una serie di link in grado di scatenare le seguenti funzioni:

- passa alla modalità di modifica per la riga selezionata;
- salva le modifiche;
- torna alla visualizzazione di tutti i dettagli.

A parte le solite proprietà condivise con gli altri control, quelle proprie della colonna sono:

- **ButtonType** come in *ButtonColumn*, specifica di che tipo deve essere il pulsante;
- **CancelText:** il testo da associare al pulsante *"Annulla";*
- **EditText:** il testo da associare al pulsante *"Modifica";*
- **UpdateText:** il testo associato al pulsante *"Aggiorna".*

GESTIRE L'EDIT MODE

L'inserimento di questo tipo di control all'interno delle colonne del *DataGrid* fa sì che vengano automaticamente aggiunti i pulsanti in grado di rendere possibile la modifica, ma quando il *DataGrid* passa in *Edit Mode* (ovvero in modalità di modifica di una riga) quello che succede è che vengono invocate le funzioni associate ai vari stati appena menzionati. Per fare sì che l'*EditCommandColumn* possa richiamare le funzioni corrette, nel nostro *DataGrid* dovremo specificare gli handler associati agli eventi stessi, in questo modo:

```
<ASP:dataGrid...
  OnEditCommand="onEditDataGrid"
  OnUpdateCommand="onUpdateDataGrid"
```

```
  OnCancelCommand="onCancelDataGrid"
/>
```

OnEditCommand, OnUpdateCommand, OnCancelCommand si verificano rispettivamente alla pressione dei tati *"Modifica", "Aggiorna", "Annulla"*. La fase di cancellazione va gestita pertanto con un button separato, sfruttando ad esempio una *ButtonColumn*. All'interno delle classi dovremo recuperare le informazioni inserite nelle *BoundColumn* (oppure nei template) e per farlo dovremo arrivarci usando la posizione dei controls nella griglia, recuperando i valori contenuti nelle textbox generate dinamicamente:

```
string nome = ((TextBox) e.Item.Cells[1].Controls[0]).Text;
```

In questo caso punteremo alla cella 1 (la seconda) ed al *control 0* (il primo) della riga selezionata ed il discorso è esattamente identico per tutte le altre colonne della griglia. Come si vede dall'esempio facciamo il casting del control in uno di tipo *TextBox*, che è poi quello che ASP .NET crea automaticamente per noi. Nel caso di colonne personalizzate che usano template, ad esempio con *DropDownList*, andrà fatto il casting nel tipo utilizzato. Una guida passo-passo alla creazione di un *DataGrid* con tanto di modifica è visibile nel box qui sotto.

PERSONALIZZAZIONE GRAFICA DEL DATAGRID

Il *DataGrid* è personalizzabile quasi in ogni suo aspetto, perché alla fine come abbiamo scoperto con questi esempi ogni riga della sorgente dati è rappresentata come una riga di una tabella HTML. Analogamente a quanto visto per *DataList* e *Repater*, anche il *DataGrid*

L'AUTORE

Daniele Bochicchio è il content manager di *ASPItalia.com*, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. È Microsoft ASP.NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo
http://blogs.aspitalia.com/daniele/

UN DATAGRID CON MODIFICA IN 5 PASSI

STRUTTURA DEL DATAGRID

```
<ASP:BoundColumn DataField="title_ID" ReadOnly=
                          "True" HeaderText="ID" />
<ASP:BoundColumn DataField="title"
                    HeaderText="Titolo del libro" />
<ASP:BoundColumn DataField="price"
                             HeaderText="Prezzo" />
<ASP:TemplateColumn HeaderText="Note">
<ItemTemplate>
  <%#DataBinder.Eval(Container.DataItem, "notes")%>
</ItemTemplate><EditItemTemplate>
    <asp:Textbox id="notes" runat="server" Rows=
   "5" Columns="50" textMode="MultiLine" Text='<%#
    DataBinder.Eval(Container.DataItem, "notes")%>'/>
  </EditItemTemplate></ASP:TemplateColumn>
```

1 **Andremo a modificare la struttura del *DataGrid* in modo che supporti i template anche in fase di edit. Poiché usiamo la tabella *Titles* contenuta nel database *pubs* che si trova installato con Sql Server o MSDE, ci baseremo su questo esempio, ma quanto esposto vale anche per database diversi o per Access (cambiando le classi utilizzate).**

AGGIUNGERE I LINK

```
<ASP:EditCommandColumn
  HeaderText="Modifica"EditText="Modifica"
  UpdateText="Aggiorna" CancelText="Annulla"
/>
```

2 **Le *BoundColumn* si occupano di creare in automatico la texbox in fase di modifica, mentre le *TemplateColumn* accettano, come abbiamo fatto, un template personalizzato.
È infine necessario aggiungere una colonna particolare, di nome *EditCommandColumn,* che mostrerà i link alle funzionalità che andremo ad implementare.**

DEFINIRE LE FUNZIONI

```
<asp:datagrid ..
DataKeyField="title_ID"
OnEditCommand="onEditDataGrid"
OnUpdateCommand="onUpdateDataGrid"
OnCancelCommand="onCancelDataGrid"
/>
```

3 **A questo punto il *DataGrid* necessita solo della definizione delle funzioni associate a questi comandi.**

COSA FANNO LE FUNZIONI?

```
// evento di pressione del pulsante Modifica
void onEditDataGrid (Object src,
                    DataGridCommandEventArgs e) {
  // cambio l'indice
  dg.EditItemIndex = e.Item.ItemIndex;
  bindData();}
// evento di pressione del pulsante Annulla
void onCancelDataGrid (Object src,
                    DataGridCommandEventArgs e) {
  // imposto l'indice a -1, ovvero non seleziono nulla
  dg.EditItemIndex = -1;
  bindData();
}
```

4 **Le funzioni dovranno, rispettivamente, preparare il *DataGrid* per l'edit mode, confermare o annullare la modifica. Negli ultimi due casi si tratta solo di variare la proprietà *EditItemIndex* del *DataGrid* che indica la riga selezionata.**

supporta un insieme di stili che si rifanno alle *TemplateColumns:*

- *AlternatingItemStyle*
- *ItemStyle*
- *EditItemStyle*
- *HeaderStyle*
- *FooterStyle*

Possiamo intervenire sugli stili oppure su base generale (per tutta la griglia), attraverso alcune proprietà il cui utilizzo è molto semplice. Di seguito sono riportate quelle più utilizzate:

- **BackColor**: indica il colore di sfondo della cella;
- **Font:** indica il font utilizzato;
- **Width:** imposta la dimensione della cella;
- **HorizontalAlign:** indica come deve essere allineata la tabella e può assumere i valori *Left, Right, Center* e *NotSet;*

In questo modo variamo lo stile su base generale:

```
<asp:DataGrid runat="server"
        BackColor="white" Width="100%"
        HorizontalAlign="Left"
        Font-Bold="True" Font-Name="Verdana"
        Font-Size="10pt" />
```

Nel caso in cui invece volessimo personalizzare il *DataGrid* in modo che il colore delle righe si alterni, ci basterà specificare:

```
<asp:DataGrid runat="server">
        <AlternatingItemStyle BackColor="red" />
        <ItemStyle BackColor="yellow" />
...
</asp:DataGrid>
```

L'effetto sarà quello di avere le righe di colore giallo e rosso, alternate tra di loro. Ovviamente per personalizzare *footer* o intestazione della griglia, piuttosto che il record in modifica, basta agire sui rispettivi *ItemStyle*.
Se invece si opta per l'utilizzo dei CSS, esista una proprietà *CssClass* che permette di specificare l'attributo *class* corrispondente alla HTML.

L'OGGETTO SQLCOMMAND

```
void onUpdateDataGrid (Object src,
                DataGridCommandEventArgs e) {
  string ID = dg.DataKeys[
                  e.Item.ItemIndex].ToString();
  string notes = ((TextBox)
                e.Item.FindControl("notes")).Text;
  string title = ((TextBox) e.Item.Cells[1]
                          .Controls[0]).Text;
  string price = ((TextBox) e.Item.Cells[2]
                          .Controls[0]).Text;
  string sql = "UPDATE titles SET title = '" +
       title.Replace("'", "''") + "', " + " price = " +
     Convert.ToDouble(price, System.Globalization
 .CultureInfo. InvariantCulture) + ", " + " notes =
     '" + notes.Replace("'", "''") + "' " + " WHERE
       title_ID = '" + ID.Replace("'", "''") + "' ";
  // esecuzione della query
  using (SqlConnection conn = new
                  SqlConnection(connstring))
  { conn.Open();
    SqlCommand objcommand = new
                     SqlCommand(sql, conn);
    objcommand.ExecuteNonQuery(); }
  // deseleziono record corrente ed aggiorno il
                                     DataGrid
  dg.EditItemIndex = -1;
  bindData(); }
// evento di pressione del pulsante Annulla
void onCancelDataGrid (Object src,
                DataGridCommandEventArgs e) {
  // imposto l'indice a -1, ovvero non seleziono nulla
  dg.EditItemIndex = -1;
  bindData();}
```

5 **La parte più complessa è invece associata alla modifica, perché è necessario andare a recuperare i valori dalle TextBox ed usando un oggetto SqlCommand, eseguire la query di modifca sul database. Il codice è tutto commentato. Salviamo e richiamiamo la pagina nel nostro browser.**

COSA METTERE IN DATAFORMATSTRING

Per formattare un campo prelevato da una fonte, ad esempio un decimal o una data, è possibile utilizzare questo formato particolare. Eccone alcuni esempi:

- **{0:d}: data breve**
- **{0:D}: data lunga**
- **{0:C}: formato valuta**
- **{0:D}: formato decimale**
- **{0:E}: formato scientifico**
- **{0:P}: formato percentuale**

Una lista completa (con altri pattern) è disponibile nell'articolo su http://www.aspitalia.com/articoli/aspplus/formattazione.aspx Da notare che i pattern possono essere specificati attraverso *Databinder.Eval* (come terzo parametro), oppure all'interno delle proprietà *DataFormatString* di una *BoundColumn*, come già specificato.

CONCLUSIONI

Modificare dati con ASP.NET diventa molto semplice grazie all'utilizzo dei *Data Controls* e del *DataGrid* in particolare.
Anche se all'inizio può sembrare un po' difficoltoso, questo approccio alla modifica consente di costruire facilmente griglie anche complesse.

Daniele Bochicchio

Come ti gestisco gli eventi

Impareremo come la programmazione ad oggetti possa influenzare la gestione degli eventi, pressione dei tasti, movimento del mouse, e anche come gli oggetti comunicano fra loro

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste

ActionScript 1.0 ed una discreta padronanza nell'utilizzo di Flash Mx e ActionScript 2.0.

Software

Flash Mx 2004 Porfessional

Impegno

Tempo di realizzazione

Flash è lo strumento principe sul web per la creazione di applicazioni interattive in grado di reagire in maniera diversa in base alle scelte operate dall'utente finale ed è per sua natura basato su eventi quali ad esempio la pressione dei pulsanti, il caricamento di un clip filmato, ecc.
Anche il codice scritto all'interno delle classi è in grado di gestire gli eventi nativi di Flash e inoltre permette ai programmatori di creare dei motori personalizzati per la loro gestione. Tra le nuove classi introdotte con ActionScript 2.0, la classe *EventDispatcher*, oltre ad essere utilizzata per gestire le interazioni con i nuovi components, ci mette nelle condizioni di creare eventi personalizzati e di mettere quindi in comunicazione tra loro diverse classi.
Vediamo attraverso un esempio come utilizzare l'*EventDispatcher* per gestire un piccolo motore di calcolo composto da due classi, una che si occupa della visualizzazione dei risultati ed una che invece si occupa dell'esecuzione delle operazioni matematiche.
Creiamo un nuovo *.fla (event.fla)* e, sempre nella cartella *drawing*, due file *.as* (*Output.as* e *Calculator.as*) che, come abbiamo detto, si occuperanno rispettivamente della visualizzazione dei dati e dell'elaborazione degli stessi.
La dichiarazione della classe *Calculator* deve essere preceduta dall'import della classe *EventDispatcher* e seguita dalla dichiarazione di tre membri pubblici che vengono usati di default da questa ultima

```
import mx.events.EventDispatcher;
class drawing.Calculator
{
   // metodi del gestore degli eventi
   public var dispatchEvent:Function;
   public var addEventListener:Function;
   public var removeEventListener:Function;
```

La dichiarazione di questi membri è fondamentale per il corretto funzionamento della classe *EventDispatcher*. Continuiamo ad esaminare il codice racchiuso nella classe tenendo presente che dovremo definire tre metodi che utilizzeremo poi per calcolare l'area di un triangolo, l'area di un cerchio e il suo perimetro. Partiamo dal costruttore che dovrà contenere una semplice operazione di inizializzazione

```
function Calculator()
{
   EventDispatcher.initialize(this);
}
```

Questo metodo statico della classe *EventDispatcher* ci consente di utilizzare i suoi metodi dalla classe *Calculator* e in particolare il *dispatchEvent* con il quale comunicheremo alla classe *Output* che "qualcosa è accaduto".
Il metodo *triangleArea*, partendo dalla base e dall'altezza del triangolo passati come argomenti, eseguirà i calcoli necessari, ovvero base moltiplicato altezza diviso due, e notificherà che il calcolo è avvenuto alla classe *Output*

```
public function triangleArea(b:Number, h:Number)
{
   var valueExec:Number = (b*h)/2;
   var obj:Object = {};
   obj.type = "resultData";
   obj.target = this;
   obj.data = {result: valueExec};
   this.dispatchEvent(obj);
}
```

L'oggetto *Object obj* lo utilizzeremo per defi-

nire il nome dell'evento e i dati che devono essere trasmessi attraverso questo.
Gli altri due metodi funzioneranno in maniera analoga operando però calcoli differenti

```
public function circleArea(r:Number):Void
{
   var valueExec:Number = Math.PI*r*r;
   .............................................................
}
public function circlePerimeter(r:Number):Void
{
   var valueExec:Number = Math.PI*2*r;
   .............................................................
}
```

Oltre all'operazione matematica il codice conterrà il meccanismo di "trasmissione" dell'evento che abbiamo visto nel metodo *triangleArea*.
La classe *Calculator* è completa, analizziamo la classe *Output* e in particolare il meccanismo utilizzato per mettere "in comunicazione" le due classi.
Per far si che la classe *Output* reagisca al *dispatchEvent* che parte dalla classe *Calculator* è necessario importare questa ultima e memorizzare una sua istanza all'interno di una proprietà della classe

```
import drawing.Calculator
class drawing.Output
{
   private var calcInstance:Calculator;
```

Il costruttore è la parte di codice più complessa contenuta in questa classe infatti, attraverso i due argomenti passati, dovremo far si che la classe "capisca" quale operazione eseguire e quali parametri passare ai metodi della classe *Calculator* e crearne subito una sua istanza registrandola come *eventListener* dell'evento *resultData*

```
function Output(type:String, arg:Array)
{
   calcInstance = new Calculator();
   calcInstance.addEventListener("resultData", this);
   switch(type)
   {
      case "triangleArea":
         calcInstance.triangleArea(arg[0], arg[1]);
      break;
      case "circleArea":
         calcInstance.circleArea(arg[0]);
      break;
      case "circlePerimeter":
         calcInstance.circlePerimeter(arg[0]);
      break;
   }
}
```

Per completare la definizione di questa classe è sufficiente definire un metodo privato il cui nome deve essere necessariamente uguale all'evento che vogliamo gestire

```
private function resultData(result:Object):Void
{
   trace(result["data"].result);
}
```

Il *trace* farà in modo di mostrarci nella finestra di output i risultati dei calcoli eseguiti.
Torniamo in Flash e inseriamo nel pannello delle azioni le istruzioni necessarie per importare la classe *Output* e far eseguire ad una nuova istanza di questa il calcolo dell'area di un cerchio di raggio 4

```
import drawing.Output
var test = new Output("circleArea", [4]);
```

La finestra riporterà l'operazione eseguita.

GESTIONE DELLE ECCEZIONI TRY...CATCH... TROW...FINALLY

Un'eccezione è un evento che si può riscontrare durante l'esecuzione di un programma che, in particolari condizioni, si discosta dall'esecuzione prevista per le istruzioni in esso contenute.
L'introduzione della gestione delle eccezioni in ActionScript 2.0 è stato uno degli elementi

SUL WEB

www.actionscript.it
www.risorseflash.it
www.ingenium.ws
http://www.dofactory.com/Patterns/Patterns.aspx
http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/patterns/

CLASSI ED OGGETTI

**Una classe è modello per la creazioni di oggetti (istanze) che hanno in comune tutte le proprietà e i metodi in essa definiti.
I metodi sono le "azioni" che un'istanza di una classe può compiere. Per definire un metodo è sufficiente dichiarare una funzione all'interno della classe definendone gli argomenti e l'eventuale data type del valore restituito dal metodo. I metodi possono essere public, private e static. Le proprietà di una classe si dichiarano prima del costruttore sotto forma di variabili e possono essere dichiarate come membri public, private e static della classe. L'ereditarietà è la tecnica utilizzata per organizzare classi di oggetti secondo una precisa gerarchia dove una classe, generalmente indicata come classe figlia, può ereditare tutti i metodi e la proprietà di un'altra detta classe padre.
Nella programmazione orientata agli oggetti con i termini interfaccia e implementazione si opera una distinzione tra cosa ci si deve aspettare dall'esecuzione di un metodo e come si è operato per implementarlo.**

• **ACTIONSCRIPT THE DEFINITVE GUIDE FOR FLASH MX** *Colin Moock*

• **ACTIONSCRIPT 2.0 ESSENTIALS** *Colin Moock*

• **OBJECT-ORIENTED PROGRAMMING WITH ACTIONSCRIPT 2.0** *Jeff Tapper*

• **AN INTRODUCTION TO OBJECT ORIENTED Programmino** *Timothy Bud*

più apprezzati dagli sviluppatori visto che, attraverso questo costrutto e l'uso combinato della classe *Error*, si possono gestire completamente gli errori e generare una serie di messaggi personalizzati che informano gli utenti del tipo di problema riscontrato.
La gestione delle eccezioni ci consente di scrivere un codice più funzionale che, in alcuni casi come nel caricamento di dati dinamici, può addirittura sostituire i comportamenti di default del Flash Player in caso di errori.
Vediamo con un esempio *(error.fla)* come farci riportare nella finestra di output diversi messaggi in base all'errore riscontrato.
Posizioniamo sullo stage due campi di testo dinamici, a cui assegniamo i nomi istanza *datas* e *errors*, e nei quali rispettivamente mostreremo dei dati e gli eventuali messaggi di errore ed un'istanza user del component *List*.
Inizializziamo la nostra interfaccia aggiungendo quattro elementi alla lista

```
this.user.addItem("Nome", 45);
this.user.addItem("Cognome", 22);
this.user.addItem("Età", 29);
this.user.addItem("Peso", 79);
```

Ora, attraverso la solita sintassi dei *listeners*, definiamo le azioni che devono essere eseguite ogni volta che l'utente clicca su un elemento contenuto nella lista

```
var listObj:Object = {};
listObj.change = function(eventObj)
{
  errors.text = "";
  var toTest = datas.text = eventObj
                        .target.selectedItem.data;
  // gestione dell'errore
  ................................................
};
this.user.addEventListener("change", listObj);
```

Vediamo ora come procedere per gestire l'errore all'interno dell'evento *change*

```
try
{
  if(toTest<30)
  {
    throw new Error("Età troppo bassa");
```

ATTRIBUTO STATIC E GESTIONE DEGLI EVENTI

Vediamo ora con un esempio pratico come creare un menu con una classe esterna *(menu_static.fla)* sfruttando la gestione degli eventi utilizzata dall'*EventDispatcher* e la naturale capacità di una proprietà dichiarata come static, ovvero di essere condivisa tra più istanze di una classe. Un menu contenente più voci è sempre necessario sapere quale voce è stata selezionata dall'utente in modo attivare e disattivare gli items che lo compongono.
Attraverso l'uso combinato degli eventi, di una proprietà static e della classe *Tween* creeremo un menu a più voci e le relative animazioni.

PREPARIAMO IL FLA: CLIP FILMATO E ASSOCIAZIONE DELLA CLASSE

1 **Apriamo un nuovo fla e al suo interno creiamo un clip filmato contenente un livello per le azioni, uno per un campo di testo dinamico a cui assegneremo il nome istanza voice_txt, uno con un pulsante invisibile il cui nome istanza sarà main_btn ed infine un livello con della grafica. Assegniamo come linkage al clip filmato la stringa item e associamolo alla classe StaticMenu contenuta nella cartellina classes/eventMenu**

DEFINIZIONE DELLA CLASSE STATICMENU: EVENTDISPATCHER

```
import mx.events.EventDispatcher;
class classes.eventMenu.StaticMenu
                         extends MovieClip {
  // ---------------------------------------
  // // metodi aggiunti dalla classe
                         EventDispatcher
  // ---------------------------------------
  private var dispatchEvent:Function;
  public var addEventListener:Function;
  public var removeEventListener:Function;
```

2 **Apriamo il file StaticMenu.as, importiamo la classe EventDispatcher dichiarando i metodi da essa utilizzati e definiamo la classe che estenderà i MovieClip. Il nostro scopo è ovviamente personalizzare il comportamento di un generico menu utilizzando gli eventi. Per farlo riccorreremo ai metodi dispatch**

DEFINIZIONE DELLA CLASSE STATICMENU: METODI E PROPRIETÀ

```
// ---------------------------------------
// proprietà static
// ---------------------------------------
private static var lstItems:Array;
// ---------------------------------------
// costruttore
// ---------------------------------------
public function StaticMenu()  {
  if (lstItems == undefined){
    lstItems = new Array();}
  lstItems.push(this);
  EventDispatcher.initialize(this);}
// ---------------------------------------
// metodi pubblici
// ---------------------------------------
public function
setActiveMenu(obj:StaticMenu) : Void{
  for(var i in lstItems){
    if (lstItems[i] == obj){
      lstItems[i].activate(); }else{
      lstItems[i].deactivate()}
  }
}
```

3 **All'interno della classe definiamo il costruttore, una proprietà static che altro non sarà se non un array con tutte le istanze dei clip filmato che compongono il menu e il metodo public setActiveMenu che attiverà e disattiverà tutte le voci**

```
    }
}

catch (error)
{
    if(error.message == "Età troppo bassa")
    {
        errors.text = "Non sei abbastanza grande!";
    }
}

finally
{
    trace("eseguito comunque")
}
```

Il codice inserito subito dopo il *try* viene immediatamente eseguito e, al suo interno, se vengono riscontrati errori, come ad esempio in questo caso un numero inferiore a 30, attraverso l'istruzione *throw* generiamo una nuova istanza della classe *Error* che intercettiamo nel catch.
Il codice inserito dopo il finally viene eseguito comunque.

Giorgio Natili

Fig. 1: ***L'applicazione che abbiamo realizzato è abbastanza semplice. I menu si muoveranno in sincronia a destra e a sinistra a seconda quale dei due registra l'evento Click***

DEFINIZIONE DELLA CLASSE STATICMENU: DISPATCHEVENT

```
private function activate() : Void{
    dispatchEvent({type:"onActivated",
                            target:this});}
private function deactivate() : Void {
    dispatchEvent({type:"onDeactivated",
                            target:this});
}
```

4 **Gestiamo l'attivazione e disattivazione delle voci di menu con due metodi private che eseguiranno il *dispatchEvent***

CREAZIONE DEL MENU

```
var voices:Array = ["home", "links",
            "contacts", "products", "clients"]
for (var i:Number = 0; i < voices.length;
                                    i++) {
var tmpClip:MovieClip =
attachMovie("item","menuItem" + i,
                this.getNextHighestDepth());
    tmpClip.voice_txt.text = voices[i];
    tmpClip._x = 50
    tmpClip._y = 10+ i * (tmpClip._height
                                    + 5);}
```

5 **Posizioniamo sullo stage tante istanze del clip filmato quante sono gli indici dell'array popolando il campo di testo in esse contenuto**

DEFINIZIONE DELLE FUNZIONI INTERNE AL CLIP FILMATO

```
function onActivated():Void {
    var tween:mx.transitions.Tween = new mx.transitions.Tween( this, "_x", mx
                    .transitions.easing .Elastic .easeInOut, _x, 100, .8, true);
    main_btn.enabled = false;
    var tmpObj:Object = {};
    tmpObj.onMotionFinished = function():Void{
        voice_txt._visible = true;}
    tween.addListener( tmpObj); }
function onDeactivated():Void {
    var tween:mx.transitions.Tween = new mx.transitions.Tween(this, "_x",
                    mx.transitions.easing.Elastic .easeInOut, _x, 50, .8, true);
    main_btn.enabled =  true;
    voice_txt._visible = false;}
main_btn.onPress = function(){
    setActiveMenu(this._parent);
}
```

6 **Torniamo nel fla e, nel livello azioni contenuto nel clip filmato definiamo due funzioni che, utilizzando la classe *Tween*, gestiranno lo spostamento della voce di menu ogni qual volta il pulsante invisibile rileverà l'evento *onPress*. Le funzioni avranno lo stesso nome passato come argomento del *dispatchEvent.* La prima delle due funzioni, *onActivated*, gestirà l'animazione della voce e l'evento *onMotionFinished* che renderà visibile il campo di testo dinamico solo alla fine della transizione. All'interno della seconda funzione invece non faremo altro che far tornare al suo posto il clip filmato e rendere di nuovo invisibile il testo associato alla voce di menu. Le ultime righe di codice assoceranno la chiamata del metodo *setActiveMenu* della classe *StaticMenu* alla pressione del pulsante *main_btn* e renderanno invisibile il campo di testo dinamico *voice_txt. Non ci resta che eseguire il programma e controllare che tutto funzioni***

Fondamenti di Javascript

Le classi fondamentali che un buon programmatore deve assolutamente conoscere per creare qualcosa di utile. Quali sono e come funzionano

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
nessuna

Software
-

Impegno

Tempo di realizzazione

Nell'articolo precedente da un lato si era descritto il modello ad oggetti, standardizzato dal W3C, di un documento HTML che il browser mette a disposizione per interagire con la pagina, e dall'altro si era introdotta la sintassi del linguaggio facendo riferimento allo standard ECMA - 262 che, di fatto, deve essere considerato la "Reference Guide" ufficiale per il linguaggio javascript. Il paragrafo 15 dello standard ECMA dal titolo "Native ECMAScript Object" descrive quali devono essere le classi messe a disposizione da javascript e di conseguenza supportate dai browser. Tali classi vengono utilizzate tipicamente per effettuare le operazioni più comuni su array, stringhe, date,... ed in genere mettono a disposizione dello sviluppatore una serie di funzioni comuni a un po' tutti i linguaggi, senza le quali javascript risulterebbe un linguaggio povero. Tali classi non permettono l'accesso a file e, in genere, a risorse del sistema operativo del client sul quale il browser contenente il codice javascript è in esecuzione, per ovvi motivi di sicurezza. Anche il DOM HTML fa anche parte delle classi javascript; tuttavia le classi riportate in tabella sono quelle core del linguaggio, senza le quali non sarebbe possibile programmare in javascript.

I PRINCIPALI STRUMENTI DI LAVORO

Consideriamo il codice seguente:

#	Classe	Descrizione
0	**GlobalObject**	La superclasse padre di … tutto javascript
1	**Array**	Permette la gestione delle strutture di dati
2	**Boolean**	Gestione dei tipi di dati booleani
3	**Date**	Permette di lavorare con le date e l'ora
4	**Math**	Permette di lavorare con le principali funzioni matematiche
5	**String**	Mette a disposizione le principali funzioni per lavorare con le stringhe

Tabella 1: Elenco delle classi core messe a disposizione da javascript

```
<script type="text/javascript">
  // La radice quadrata di -3 ritorna il valore NaN
alert(Math.sqrt(-3));
  //Il valore di 10 elevato a 100000 ritorna il valore
                                            Infinity
alert(Math.pow(10,1000000));
  // Una variabile alla quale non è stato assegnato
                                alcun valore è undefined
var x;
alert(x);
</script>
```

L'istruzione *alert()*, produce una finestra di popup con un messaggio. Nello script abbiamo anticipato dei metodi della classe *Math()* che dettaglieremo in seguito ma che per altro sono abbastanza intuitive. La funzione *sqrt(x)* fornisce la radice quadrata di *x*. La funzione *pow(x,y)* fornisce il risultato di *x* elevato ad *y*. Se facciamo girare lo script vediamo una serie di messaggi 'strani'. In particolare:

- **alert(Math.sqrt(-3))** mostra un messaggio con la stringa *"NaN"*
- **alert(Math.pow(10,1000000));** mostra un messaggio con la scritta *"Infinity"*
- L'ultimo messaggio che compare relativo al codice:

```
var x;
alert(x);
mostra la stringa "undefined"
```

Bellissimo! Che cosa abbiamo combinato? Analizzando lo script, vediamo che il primo spezzone di codice cerca di calcolare la radice quadrata di un numero negativo, che non è calcolabile e dovrebbe mandare in errore il programmino. Invece di andare in errore, javascript riconosce che è impossibile fare il calcolo e che il risultato dell'operazione è un "non numero", la stringa *NaN = Not a Number.* Nel secondo spezzone di

codice, il risultato dell'operazione è un numero troppo grande perché possa essere gestito da javascript. Di nuovo, invece di andare in errore, javascript restituisce una stringa come risultato dell'operazione che informa che il numero è "Infinito". Nell'ultimo spezzone di codice si dichiara una variabile *x* senza per altro assegnarle nessun valore. Se si cerca di stampare il valore, javascript riporta che la variabile è *"undefined"*, vale a dire non definita. Le proprietà *NaN*, *Infinity* e *undefined* sono le prime "costanti" un po' particolari con le quali si avrà sempre a che fare quando si sviluppa in javascript. Senza entrare troppo nella teoria, questi valori sono dichiarati essere le proprietà della superclasse denominata *GlobalObject* nella *Specifica ECMA Script*. Tale specifica definisce anche una serie di metodi generali della classe *GlobalObject* che analizzeremo tra breve. Occorre però procedere prima con una breve digressione sull'utilizzo degli oggetti e delle classi in javascript…

CLASSI E OGGETTI IN JAVASCRIPT

Una classe deve essere vista come una scatola nera che espone metodi e proprietà che sono utilizzabili a livello di codice. Senza entrare in merito alla teoria della programmazione ad oggetti, ricordiamo che:

- L'oggetto è un'istanza di una classe. Una stessa classe può avere contemporaneamente più istanze, virtualmente sino a che la memoria del computer lo permette.
- La proprietà di una classe è assimilabile ad una variabile, nella quale il valore può essere scritto e/o letto.
- Il metodo di una classe è assimilabile ad una funzione, che può accettare dei valori di input e restituisce dei valori di output.

Un oggetto è, per così dire, una copia operativa della classe. Per creare un oggetto si utilizza la parola chiave *new* seguita dal costruttore dell'oggetto, come nell'esempio seguente:

```
var  d = new Date();
```

Entreremo in merito alla classe *Date()* più avanti, per ora interessa distinguere il concetto di classe da quello di oggetto. Consideriamo il seguente codice:

```
<script type="text/javascript">
1   var  d = new Date(2005,03,01);
2   var  d1 = new Date(2005,03,01);
    // Valorizzo la data con la proprietà setDate
3   d.setDate("26");
4   d1.setDate("30");
    // Stampo la data
5   document.write("Data d = " + d.getDate() + "/" +
        d.getMonth() +"/" + d.getFullYear()+ "<br>");
6   document.write("Data d1 = " + d1.getDate() + "/"
      + d1.getMonth() +"/" + d1.getFullYear()+ "<br>");
</script>
```

Le righe **1** e **2** creano due oggetti di tipo *data*, che derivano dalla stessa classe *Date()*, ma sono distinti e separati l'uno dall'altro, pur contenendo la stessa data 01/03/2005. Le righe **3** e **4** impostano due differenti giorni per le date *d* e *d1*, rispettivamente il 26 ed il 30. Le righe **5** e **6** stampano la data completa registrata dentro l'oggetto *d* e *d1*. Le date restituite in output sono differenti; per stampare le date viene utilizzato il metodo *getDate()* per recuperare il giorno della data, il metodo *getMonth()* per recuperare il mese della data ed infine il metodo *getFullYear()* per recuperare l'anno in quattro cifre. Si noti che per leggere o valorizzare ad esempio la proprietà *setDate()* ed anche per richiamare un metodo, si utilizza la notazione punto:

```
Oggetto.metodoOggetto(parametro1, parametro2,..).
Oggetto.proprietàOggetto()
```

Riprendiamo ora il nostro filo conduttore e analizziamo in dettaglio proprietà e metodi della classe *GlobalObject*.

	Proprietà	
p1)	NaN	rappresenta una entità non numerica Il valore iniziale è "NaN"
p2)	Infinity	rappresenta una entità numerica che supera le capacità di calcolo di javascript
p3)	undefined	rappresenta una entità che non è definita

Tabella 2: Possibili valori assunti da una variabile in caso di errore

METODI DELLA CLASSE GLOBAL OBJECT

La classe *GlobalObject* è un po' particolare. Non è mai necessario istanziarla ed è possibile utilizzare i suoi metodi e le sue proprietà semplicemente richiamandole, come fatto nel primo script in apertura dell'articolo.
Attenzione alle lettere maiuscole e minuscole nella definizione di queste proprietà che come al solito hanno significato. Quindi, dal punto di vista pratico, le proprietà del *Global Object* che d'ora in avanti chiameremo semplicemente costanti generali, permettono in qualche modo di effettuare dei controlli su errori eventualmente presenti nel codice.

TAINT ED UNTAINT DI NETSCAPE NAVIGATOR

Netscape Navigator nella versione 3 ha creato un modello denominato Data-tainting Security Model che implementava la funzionalità di tainting (letteralmente scomposizione) ed untainting il cui obiettivo era quello di impedire il passaggio di dati generati con javascript verso l'utente finale senza il permesso di quest'ultimo. Tale metodo non ha dato i risultati sperati ed è stato abbandonato.

IL METODO EVAL()

Il primo metodo *eval()* serve a valutare il valore delle variabili javascript a partire dal nome della variabile e più in generale ad eseguire delle stringhe di codice javascript che all'interno del codice possono essere modificate in fase di run time. Ad esempio:

```
<script type="text/javascript">
var nome_1 = "Marco";
var nome_2 = "Giacomo";
var nome_3 = "Giovanni";
for(i=1;i<=3;i++)
{ var nomeVar = "nome_" + i;
   document.write("Ciao, il mio nome e\' :" +
                                   eval(nomeVar) + "<br>");}
</script>
```

In questo esempio, le tre variabili *nome_1, nome_2* e *nome_3* hanno la caratteristica di avere la radice comune *"nome_"* seguita da un indice progressivo. La variabile *nomeVar* all'interno del ciclo *for,* viene successivamente valorizzata con la stringa *"nome_1", "nome_2"* e *"nome_3".* Attraverso l'istruzione: *eval("nome_1")* accedo al contenuto della variabile *nome_1* che vale *"Marco".* Se non si fosse utilizzata l'istruzione di *eval(),* si sarebbe ottenuto sempre la stampa del contenuto *nomeVar* che è una stringa contenente successivamente i valori: *nome_1, nome_2* e *nome_3.* A tal proposito si considerino i risultati delle seguenti istruzioni:

La radice quadrata di un numero negativo è un numero complesso, come è noto dalla matematica. Alcuni linguaggi, ad esempio Python hanno il supporto nativo di tipi immaginari. In pratica,coosì come è possibile dichiarare un numero di tipo Intero, è anche possibile dichiarare un numero di tipo immaginario. Javascript non da questa possibilità.

```
<script type="text/javascript">
var nome_1 = "Marco";
   var nomeVar = "nome_" + i;
   document.write("Ciao, il mio nome e\' : " +
      nomeVar_1 + "<br>") // Restituisce: Ciao, il mio
                                          nome è nome_1
   document.write("Ciao, il mio nome e\' : " +
             eval(nomeVar) + "<br>")   // Restituisce:
                                   Ciao il mio nome è Marco
</script>
```

La prima riga stampa il nome della variabile *nomeVar* che vale *nome_1,* ma che non è il risultato voluto. La seconda stampa il contenuto della variabile *nome_1,* che è *Marco* e che è proprio quello che si voleva ottenere. Alcuni altri esempi del comportamento del metodo *eval().*

```
// eval() senza parametri:
document.write ("La funzione eval() senza parametri
              restituisce: " + "<b>" + eval() + "</b>");
// Viene stampato: "La funzione eval() senza
                          parametri restituisce: undefined"
// eval() di un numero:
document.write ("La funzione eval(456) restituisce: " +
                              "<b>" + eval(456) + "</b>");
// Viene stampato: "La funzione eval(456) restituisce: 456"
```

Nel caso in cui la stringa che si passa al metodo *eval()* ci sia del codice javascript errato, viene restituito un errore, come nell'esempio seguente, nel quale si è utilizzata la funzione inesistente *allerta()* invece di *allert()* per inviare un messaggio di popup.

```
function EvalErrato()
{  // Eval di una istruzione javascript con errore:
   var strToEval = "allerta(\"Ciao!\")";
   eval(strToEval); }
</script>
<form name="frmEval" id="frmEval">
   <input type="button" value="eval(allerta('Ciao!')) errato"
      name="exEval" id="ExEval" onClick="EvalErrato()">
</form>
```

Premendo il pulsante sopra definito etichettato *"eval(allerta('Ciao!')) errato"* si ottiene un errore javascript. Si può realizzare una semplicissima interfaccia che può servire a testare il codice *javascript.* Il codice è il seguente:

```
<form name="frmTuoEval" id="frmTuoEval">
   <textarea name="TextArea" rows="20" cols="60">
   alert("Scrivi nella textarea tuo codice javascript
                              senza usare il tag script");
      </textarea>
   <input type="button" value="Esegui!" name=
      "exEval1" id="ExEval1" onClick="javascript:eval(
                 document.frmTuoEval.TextArea.value)">
</form>
```

IL METODO PARSEINT (STRINGA S, BASE B)

Il secondo metodo *parseInt(stringa s, base b)* serve per convertire una stringa *s* contenente una espressione numerica in un certa base *b* in un numero intero in base *10.* In particolare, *parseInt(...)* prende in input due valori. Il primo è la stringa che deve essere convertita in un intero ed il secondo, opzionale, è la base del numero (esadecimale, binaria, ottale ed in genere va bene un qualunque numero compreso tra 2 e 36) contenuto nella stringa s e che deve essere convertito in base decimale. Alcuni esempi sono:

```
stringa = "123456";
document.write("stringa = " + stringa + "<br>");
document.write("\"parseInt(stringa, 10)\" = " +
                                  parseInt(stringa, 10));
```

Questo esempio restituisce il numero intero 123456 in base 10, che coincide con se stesso.

```
stringa = "123456.789";
document.write("stringa = " + stringa + "<br>");
document.write("\"parseInt(stringa, 10)\" = " +
```

```
                                  parseInt(stringa, 10));
```

Questo esempio restituisce il numero intero 123456 senza tenere conto dei numeri decimali.

```
stringa = "01001001110";
document.write("stringa = " + stringa + "<br>");
document.write("\"parseInt(stringa, 2)\" = " +
                                   parseInt(stringa, 2));
```

Questo esempio restituisce la conversione in decimale del numero binario 1001001110 che vale 590.

```
stringa = "0xD7";
document.write("stringa = " + stringa + "<br>");
document.write("\"parseInt(stringa, 16)\" = " +
                                  parseInt(stringa, 16));
```

Questo esempio restituisce la conversione in decimale del numero esadecimale *0xD7* che vale 215.

```
stringa = "012 1234567 e\' il numero  del mio
                                            telefono!";
document.write("stringa = " + stringa + "<br>");
document.write("\"parseInt(stringa, 10)\" = " +
                                  parseInt(stringa, 10));
```

La funzione *parseInt( )* cerca il primo carattere della stringa; se questo è numerico prosegue al secondo carattere; se anche quanto è numerico prosegue sino a che non trova un carattere non numerico.
Nell'esempio i primi tre caratteri sono numerici ed il quarto è uno spazio, perciò la funzione di ricerca dei numeri di *parseInt( )* si interrompe. Otterremo 12, in quanto lo zero iniziale è eliminato perchè non significativo.

```
stringa = "Una stringa con numeri: 1, 2 345,...";
document.write("stringa = " + stringa + "<br>");
document.write("\"parseInt(stringa, 10)\" = " +
                                  parseInt(stringa, 10));
```

La funzione *parseInt( )* ritorna *NaN* in quanto il primo carattere della stringa e non numerico e la ricerca di valori numerici da parte della funzione *parseInt()* si ferma immediatamente.

```
stringa = "45678954";
document.write("stringa = " + stringa + "<br>");
document.write("\"parseInt(stringa, 2)\" = " +
                                   parseInt(stringa, 2));
```

L'ultimo esempio ritorna anche *NaN* in quanto *45678954* non è un numero in formato binario, che deve solo contenere il valore *0* ed *1*. Quindi la funzione *parseInt()* restituisce *NaN*. Nel caso in cui non è inserito il parametro opzionale base del metodo *parseInt()* (o che questo valga zero), la procedura utilizzata dalla funzione per determinare la base in cui il numero è rappresentato è:

1) Se la stringa inizia per i numeri da 1 a 9, la rappresentazione del numero è considerata essere decimale, quindi *base=10*.
2) Se la stringa inizia per *0x* oppure *0X*, la rappresentazione del numero è considerata essere esadecimale, quindi *base=16*.
3) Se la stringa inizia per *0*, la rappresentazione del numero è considerata essere ottale, quindi *base= 8*.

Senza specificare la base, nell'ultimo esempio, la funzione *parseInt()* avrebbe interpretato correttamente il numero *45678954* in base decimale e lo avrebbe stampato correttamente. È in ogni modo sempre consigliato specificare il valore della base di partenza del numero che si sta convertendo, onde evitare problemi sulla corretta interpretazione della stessa. Lasciamo al lettore eseguire altre verifiche sul comportamento del metodo *parseInt(...)*.

CONCLUSIONI

Nell'articolo abbiamo discusso dei metodi e delle proprietà del cosiddetto *Global Object*, che possono essere referenziate senza l'utilizzo della parola-chiave *new* in qualsiasi punto del codice. Pur citandole, non abbiamo trattato le proprietà ed i metodi concernenti la gestione delle URL, che richiedono un articolo a parte. Nel prossimo articolo analizzeremo in dettaglio proprio questi metodi e proprietà.

Danilo Berta

LA CLASSE GLOBAL OBJECT

Abbiamo detto che la classe Global Object, a essere precisi, non è neanche una classe, non potendone creare delle istanze mediante la parola chiave new e non avendo essa un costruttore. Chi tra i lettori conosce già javascript potrebbe avere da obiettare. Infatti l'istruzione javascript:

```
mioOggetto = new
             Object()
```

è perfettamente valida; l'istruzione:

```
alert(mioOggetto)
```

restituisce una finestra di popup con la scritta *[object Object]*. Attenzione! Non fate confusione tra il Global Object e la classe Object. Non è un gioco di parole. Nella classe Global Object sono definiti una serie di metodi e di proprietà comuni e trasversali (al top level, se così possiamo dire) che possono essere utilizzati direttamente in qualunque parte del codice senza la necessità di creare istanze della classe, che di fatto non si possono creare. La classe *Object()* è l'origine per tutte le altre classi javascript ed essendo una classe a tutti gli effetti, le sue istanze devono essere create utilizzando la parola-chiave *new* e ed il suo costruttore che è proprio *Object()*. Di questa classe parleremo diffusamente negli articoli successivi.

La grafica arriva sul Cellulare

Symbian OS, il sistema operativo di molti cellulari dispone di funzionalità avanzate per la gestione di interfacce grafiche Scopriamo gli strumenti fondamentali per programmarle

In questo articolo ci divertiremo a scrivere una semplice applicazione per un cellulare Symbian. La particolarità del nostro approccio è che parleremo addirittura di "Interfaccia Grafica"!
Una novità per la programmazione dei cellulari, che come tutti sanno non dispongono di grandi display e di risorse particolarmente generose. La nostra applicazione invece mirerà proprio a costruire una grafica accattivante per un applicazione che dovrà funzionare su cellulare.

UIKON E AVKON

Il motore grafico di Symbian OS è chiamato *Uikon*. Esso non è solo un framework per lo sviluppo di applicazioni ma anche un ricco contenitore di componenti standard come dialog, editor di testo ed etichette. Utilizzare elementi grafici pronti anziché implementarli da zero ci permette un notevole risparmio di tempo e fatica. Il punto fondamentale è che diversi dispositivi hardware offrono differenti piattaforme di sviluppo evolvendo concetti e funzioni già presenti nel sistema sottostante. *Avkon* è l'estensione a *Uikon* per la Serie 60 di casa Nokia. Ponendosi come layer aggiuntivo, *Avkon* ridisegna l'ambiente grafico e offre funzionalità nuove che seguono tutte uno stesso stile, ampliando la possibilità di personalizzazione. Un design, insomma, caratteristico solo della Nokia.

PER INIZIARE

È necessario installare l'SDK relativo al proprio cellulare. Useremo la Developer Platform 1.2 per la Serie 60 della Nokia, reperibile al seguente url: http://www.forum.nokia.com/main/0,6566,034-4,00.html. Ciò ci permetterà di effettuare il build dell'applicazione su modelli come il 7650 o il 3660. La versione 2.0 della piattaforma offre invece più funzionalità ed è supportata da tutta una gamma di dispositivi (6600, 6630, 6670, etc.). Indispensabile installare un'interprete perl. ActivePerl per Windows è disponibile all'indirizzo http://www.activeperl.com/Products/ActivePerl/. Come ultima cosa abbiamo bisogno di Microsoft Visual C++ versione 6.0 o superiore, ricordandoci, in fase di installazione, di spuntare la checkbox per il settaggio delle variabili d'ambiente. Installato il software, possiamo creare un nuovo progetto tramite il Nokia Application Wizard, presente come applicativo nell'SDK 2.0 o come template per Visual C++ nell'SDK 1.2.

PARTI DEL PROGRAMMA

Come forse tutti sappiamo l'input-output su un dispositivo mobile, ovvero ciò che l'utente inserisce e ciò che l'utente vede a schermo, viene gestito tramite piccoli dialog, bottoni, icone ed altri elementi grafici. Creare un software significa, quasi esclusivamente, progettare una serie di componenti comunicanti tra loro, e nel nostro caso uno di questi deve essere l'interfaccia grafica.
Un programma per Symbian OS è essenzialmente composto da quattro parti:

- La classe principale
- Il Document
- L'interfaccia utente
- La vista

Ciascuno di questi elementi è rappresentato da una classe.
La classe principale è il punto d'accesso per l'esecuzione del programma. Essa estende *CAknApplication* e ha il compito di creare il *Document*, un oggetto di una classe derivata da *CAknDocument*. In genere il *Document*, sebbene necessario, conserva solo il riferimento all'interfaccia utente, svolgendo quindi una funzione di "ponte".

Per le applicazioni che utilizzano i file, il *Document* si rivela un'utile strumento per il salvataggio e il recupero dei dati appartenenti al programma. L'interfaccia utente, derivata da *CAknAppUI,* svolge il ruolo di gestore, operando cambiamenti di vista in seguito a determinati eventi provenienti dal sistema. Ciò che manipola le informazioni a schermo è la vista. I modi di implementare una vista sono prevalentemente due: tramite l'estensione di *CAknView* e tramite i *Dialog.* Creare oggetti di una classe derivata da *CAknView* ci permette di far coesistere sul display diverse schermate, ognuna con i suoi componenti grafici, e lo switch tra una schermata e l'altra è possibile grazie ad appositi controlli (menu, bottoni, etc.) o alle "tab".
Tuttavia se l'applicazione è rappresenta come un albero, in cui nei nodi (*Dialog*) si decide il cammino successivo, la seconda tecnica è quella più adatta.

DALLA TEORIA ALLA PRATICA

Per iniziare creiamo un progetto base con l'*AppWizard* e chiamiamolo *Leonardo.*
Ricordiamoci di spuntare la checkbox *Support view* architecture.
Posizionandoci in *c:\Leonardo\src* osserviamo la presenza di alcuni file, ognuno di essi costituisce una parte del programma. Le due viste, *LeonardoView* e *LeonardoView2,* utilizzano due classi per posizionare gli elementi sullo schermo, *LeonardoContainer* e *LeonardoContainer2* rispettivamente. I *Container* estendono la classe *CcoeControl* e implementano le funzioni utili per la visualizzazione.
La seguente porzione di codice, presente nei due costruttori, crea un'etichetta di testo e le assegna il *Container* di riferimento, ovvero l'istanza corrente **this*:

```
iLabel = new (ELeave) CEikLabel;
iLabel->SetContainerWindowL( *this );
iLabel->SetTextL( _L("Prima vista") );
```

La dimensione della label viene settata all'interno della funzione *SizeChanged():*

```
iLabel->SetExtent( TPoint(10,10), iLabel->
                                        MinimumSize() );
```

Il primo parametro rappresenta la posizione all'interno dell'area occupata dal *Conteiner* e il secondo ne comunica la grandezza. Per passare da una vista all'altra si usano le frecce sinistra e destra del pulsante di navigazione. Il cambio di vista viene effettuato richiamando le funzioni *DoDeactivateL()* della vista uscente e *DoActivateL()* di quella entrante.
Il codice scritto dall'AppWizard in *LeonardoAppUi.cpp* associa la pressione della freccia all'esecuzione delle funzioni interessate.
DoActivateL() sfrutta il seguente codice per visualizzare il *Container:*

```
AppUi()->AddToStackL( *this, iContainer );
```

Inoltre notiamo la presenza di queste righe

```
iContainer = new (ELeave) CLeonardoContainer;
iContainer->SetMopParent(this);
iContainer->ConstructL( ClientRect() );
```

che ricreano l'istanza del *Container* ad ogni attivazione della vista e che possono essere spostate nel costruttore. Senza perdere generalità sfruttiamo il codice esistente e aggiungiamo ciò che ci serve.
Entrambi i *Container* devono avere la seguente funzione:

```
void setLabelText(const TDesC& aText)
{
   iLabel->SetTextL(aText);
   SizeChanged();
}
```

COME FUNZIONA?

Per avviare un'applicazione il framework richiama tre funzioni in sequenza presenti nella classe principale: *NewApplication(), AppDllUid()* e *CreateDocumentL().* La prima funzione restituisce un'istanza della propria classe e la seconda un identificativo (Uid). L'Uid permette di controllare se il programma è già in esecuzione. *CreateDocumentL()* crea e restituisce il *Document.* La creazione del *Document* comporta l'inizializzazione dell'interfaccia utente e quindi delle viste (o dei *Dialog*). Una volta completati i passi preliminari il thread per la gestione degli eventi prende vita, il suo nome è *CONE.*
Il compito del software a questo punto è quello di rispondere in maniera corretta agli stimoli provenienti dal sistema. Eventi provenienti da bottoni, menu o altri controlli, così come eventi per il refresh del display, devono permettere l'esecuzione dei relativi task. Infine il cambio di vista viene effettuato dall'interfaccia utente in seguito a eventi prestabiliti, richiamando oppurtunamente le funzioni *DoActivateL()* e *DoDeactivate()* delle viste interessate.

Essa ci servirà per cambiare la stringa visualizzata sul display. Le due viste gestiscono l'input utente in maniera diversa, la prima tramite un oggetto *CaknTextQueryDialog* e la seconda tramite una lista di popup.
CLeonardoView1 deve gestire l'evento nel seguente modo, all'interno di *HandleCom-*

CHE COS'È UN UID?

L'Uid è un numero casuale unico che identifica un programma ed è generato in fase di creazione del progetto.

DOVE STANNO I FILE?

La directory di lavoro deve risiedere nella stessa partizione dove è installato l'SDK.

mandL(TInt aCommand):

```
switch ( aCommand )
{
  case ELeonardoCmdChange:
  {
    TBuf<128> textData;
    CAknTextQueryDialog * dlg = new (ELeave)
                      CAknTextQueryDialog(textData,
                        CAknQueryDialog::ENoTone);
    if (dlg->ExecuteLD(R_LEONARDO_QUERY))
      iContainer->setLabelText(textData);
    break;
  }
}
```

STILI GRAFICI

Sulla piattaforma Symbian è possibile cambiare l'aspetto di tutti i componenti grafici presenti, conferendo al display un look molto accattivante. Ogni applicazione può sfruttare uno specifico look and feel, come avviene per le Java Swing. Inoltre è possibile usare il "Series 60 Theme Studio" http://www.forum.nokia.com/main/0,6566,034-301,00.html per velocizzare la creazione dei propri temi.

Il codice crea un dialog contenente un box per l'inserimento di testo. *EleonardoCmdChange* è definito in *c:\Leonardo\data\leonardo.hrh* ed è un intero che identifica in maniera univoca l'evento. Nonostante il codice sia abbastanza semplice, *R_LEONARDO_QUERY* necessita di un approfondimento. Se osserviamo bene i sorgenti notiamo che nulla è scritto riguardo l'estetica dei componenti. Questo perché tutto ciò che riguarda gli elementi grafici è contenuto in un file apposito, *c:\Leonardo\data\Leonardo.rss* nel nostro caso. Analizzandone il contenuto troviamo blocchi del seguente tipo:

```
RESOURCE nome_struct nome_risorsa
{
  campo=valore;
  campo=valore;
  ...
}
```

dove nome_struct sono delle struct ben definite in *avkon.rh* (es. *MENU_BAR, MENU_PANE, DIALOG,* etc.). Implementare una nuova vista significa sostanzialmente aggiungere all'interno di questo file una risorsa di tipo *AVKON_VIEW* (naturalmente il riferimento della nuova vista deve essere gestito all'interno di *LeonardoAppUi.h*). Scendere troppo nei dettagli sarebbe non solo tedioso ma anche fuori dagli obiettivi dell'articolo. L'SDK inoltre è corredato di un'ottima documentazione al riguardo.

CLeonardoView2 contiene invece la seguente funzione, richiamata all'interno del blocco *case EleonardoCmdChange:*

```
void CLeonardoView2::ShowPopupListL()
{
  // Crea la listbox.
  CEikTextListBox * list = new (ELeave)
                    CAknSinglePopupMenuStyleListBox;
  CleanupStack::PushL(list);
  // Crea la lista di popup.
  CAknPopupList* popupList = CAknPopupList::NewL(
                  list, R_AVKON_SOFTKEYS_OK_BACK,
                    AknPopupLayouts::EMenuWindow);
  popupList->SetTitleL(_L("Seleziona un elemento:"));
  CleanupStack::PushL(popupList);
  // Inizializza la listbox.
  list->ConstructL(popupList,
```

1 Installiamo il software: *Nokia Serie 60 Developer Platform 1.2, ActivePerl* e *Microsoft Visual C++ 6.0* o superiore.

2 Decomprimiamo il file *Leonardo.rar* nella partizione in cui abbiamo installato gli strumenti necessari.

3 Analizziamo i sorgenti, presenti nella directory *Leonardo/src*, mirando alla comprensione dei punti chiave specificati nell'articolo.

```
                    CEikListBox::ELeftDownInViewRect);
    list->CreateScrollBarFrameL(ETrue);
    list->ScrollBarFrame()->SetScrollBarVisibilityL(
    CEikScrollBarFrame::EOff, CEikScrollBarFrame::EAuto);
    // Ricava gli elementi dal file di risorsa.
    CDesCArrayFlat* items = iCoeEnv->
                                ReadDesCArrayResourceL(
                            R_LEONARDO_MENU_ITEMS);
    CleanupStack::PushL(items);
    // Aggiunge gli elementi al modello della listbox.
    CTextListBoxModel* model = list->Model();
    model->SetItemTextArray(items);
    model->SetOwnershipType(ELbmOwnsItemArray);
    CleanupStack::Pop();
    // Visualizza la lista ed esamina il valore selezionato.
    TInt popupOk = popupList->ExecuteLD();
    CleanupStack::Pop();          // popuplist
    if (popupOk)
    {
      TInt index = list->CurrentItemIndex();
      if (index == 0) iContainer->
                    setLabelText(_L("Primo elemento"));
      else if (index == 1) iContainer->
                  setLabelText(_L("Secondo elemento"));
      else iContainer->setLabelText(_L("Terzo elemento"));
    }
    CleanupStack::PopAndDestroy();  // list
}
```

Il codice è abbastanza semplice. La lista di popup, oggetto della classe *CaknPopupList*, attinge i propri dati da un modello. Il modello è popolato con la risorsa *R_LEONARDO_MENU_ITEMS*, appartenente al file di risorsa, che è di tipo *ARRAY*:

```
RESOURCE ARRAY r_leonardo_menu_items
{
    items =
    {
      LBUF { txt = "Primo Elemento"; },
      LBUF { txt = "Secondo Elemento"; },
      LBUF { txt = "Terzo Elemento"; }
    };
}
```

CONCLUSIONI

Nel corso dell'articolo abbiamo utilizzato solo una piccola parte degli strumenti a disposizione, concentrando la nostra attenzione, più che altro, sul funzionamento del framework. Capire i meccanismi che stanno alla base è ciò che ci permette di essere flessibili in diversi contesti. D'altra parte, saper scorrere un elenco di API non è la stessa cosa che saperle implementare da zero.

Antonio Trapani

UN'OCCHIATA AL DISPLAY

Il display dei cellulari Symbian può essere sostanzialmente suddiviso in tre aree: il pannello di stato, il pannello principale e il pannello di controllo. Il pannello di stato offre informazioni sul sistema come la carica della batteria o l'ora corrente. In genere esso contiene il nome del programma in esecuzione e la sua icona, ma è possibile modificarlo. Il pannello principale è l'area effettiva dove il software prende posto. Infine il pannello di controllo permette di compiere delle azioni ed è accessibile dai due pulsanti funzione posti, di solito, ai lati del navigatore (le quattro frecce per intenderci).

4 **Posizioniamoci in *Leonardo/group* e compiliamo il progetto tramite i seguenti comandi: *bldmake bldfiles, abld build thumb urel.***

5 **Creiamo il file di installazione. Posizioniamoci in *Leonardo /install* e digitiamo *makesis Leonardo .pkg.***

6 **Dopo aver trasferito e installato il software sul dispositivo, eseguiamolo e verifichiamo la funzionalità dei controlli grafici.**

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

RIAVVIARE IL PC

Poche righe di codice per resettare il sistema. Da usare con cautela!

```
Public Declare Function ExitWindowsEx Lib "User32" Alias
"ExitWindowsEx"(ByVal
uFlags as Long,ByVal dwReserved as Long) as Long
Public sub Form_Load()
Call ExitWindowsEx(EWX_REBOOT,0)
End Sub
```

COME SALVARE POSIZIONE E DIMENSIONE DI UNA FORM

Di seguito trovate due funzioni SaveSetting e GetSetting che consentono di salvare la posizione e le dimensioni di una form. L'utilizzo più tipico è quello di far ritrovare all'utente una form così come l'aveva lasciata alla chiusura. Il codice che segue va scritto in un apposito modulo:

```
Public Sub FormPosition_Get(F As Form)
' Retrieve Form F's position from an
' ini/reg file and position it
' accordingly
Dim buf As String
Dim l As Integer, t As Integer
Dim h As Integer, w As Integer
Dim pos As Integer

buf = GetSetting(app.EXEName, _
        "FormPosition", F.Tag, "")
If buf = "" Then
        ' defaults to centering the form
        F.Move (Screen.Width - F.Width) \ _
                2, (Screen.Height - F.Height) \ 2
Else
        ' extract l,t,w,h and move the form
        pos = InStr(buf, ",")
        l = CInt(Left(buf, pos - 1))
        buf = Mid(buf, pos + 1)
        pos = InStr(buf, ",")
        t = CInt(Left(buf, pos - 1))
        buf = Mid(buf, pos + 1)
        pos = InStr(buf, ",")
        w = CInt(Left(buf, pos - 1))
        h = CInt(Mid(buf, pos + 1))
        F.Move l, t, w, h
End If
End Sub
Public Sub FormPosition_Put(F As Form)
' Write form F's top,left,height and
' width properties to the reg/ini file
' for the application
Dim buf As String
buf = F.left & "," & F.top & "," & _
        F.Width & "," & F.Height
SaveSetting app.EXEName,_
        "FormPosition", F.Tag, buf
End Sub
```

Dagli eventi Load e Unload della form, potete richiamare le funzioni come segue:

```
Sub Form_Load()
        FormPosition_Get Me
End Sub
Sub Form_Unload()
        FormPosition_Put Me
End Sub
```

RENDIAMO LA VITA DIFFICILE AI CRACKER

Caricando le variabili del nome utente e seriale attraverso l'evento OnChange, e calcolando il seriale solo in un secondo momento, attraverso l'evento *OnClick* di un bottone, renderemo impossibile *brekkare* su API come *hmemcpy, GetDlgItemTextA* etc... Questo ci metterà al riparo.

Tip inviato da ev0

```
procedure TForm1.LabeledEdit1Change(Sender: TObject);
begin
Nome := LabeledEdit1.Text
end;
procedure TForm1.LabeledEdit2Change(Sender: TObject);
begin
Seriale := LabeledEdit2.Text
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
Calcolo := Nome[1]+Nome[2]+Nome[4]+Nome[7];
IF Seriale = Calcolo
then
MessageDlg('Esatto',mtCustom ,[mbOK], 0)
else
MessageDlg('Sbagliato!',mtCustom ,[mbOK], 0);
Edit1.Text := Calcolo
end;
```

COME GESTIRE LA DATA IN JAVA

Utilizzando Swing un esempio di spinner è il seguente

```
import javax.swing.*;
import javax.swing.event.*;
import java.text.*;
import java.awt.*;
import java.util.*;
public class Spinner {
 public static void main (String args[]) throws Exception {
  JFrame frame = new JFrame("Spinner");
  frame.setDefaultCloseOperation(3);
  String[] months = new DateFormatSymbols().getMonths();
  SpinnerDateModel model2 = new SpinnerDateModel();
  model2.setCalendarField(Calendar.WEEK_OF_MONTH);
  JSpinner spinner2 = new JSpinner(model2);
  JSpinner.DateEditor editor2 = new JSpinner.DateEditor(
   spinner2, "MMMMM dd, yyyy");
  spinner2.setEditor(editor2);
  frame.getContentPane().add(spinner2, BorderLayout.SOUTH);
  ChangeListener listener = new ChangeListener()
  {
   public void stateChanged(ChangeEvent e)
   {
    SpinnerModel source = (SpinnerModel)e.getSource();
    System.out.println("The value is: " + source.getValue());
   }
  };
  model2.addChangeListener(listener);
  frame.pack();
  frame.show();
 }
}
```

IL TIP DEL MESE

UNA CLASSE CHE PERMETTE DI STAMPARE SULLA STAMPANTE CON POCHE RIGHE DI CODICE

Tip fornito dal sig. Aldo Comito

```
import java.awt.*;
import java.awt.print.*;
import java.io.*;
/* la prova è sta effettuata su un foglio A4 */
public class PrinterFast
{  String path;
   public void stampa()
  {  PageFormat pf= new PageFormat();
      int x=(int)pf.getImageableX();
                /* coordinate 0,0 del foglio */
      int y=(int)pf.getImageableY();
      path="percorso del file da stampare";
      JobAttributes jobAttr= new
                              JobAttributes();
      PageAttributes pageAttr= new
                              PageAttributes();
      Toolkit tk= Toolkit.getDefaultToolkit();
      PrintJob pj = tk.getPrintJob(new
             Frame(),"Stampa PrinterFast",
                          jobAttr,pageAttr);
      Graphics g= pj.getGraphics();
      String line;
      int countPages=1;
      try
      { File source= new File(path);
        BufferedReader buff= new
            BufferedReader(new FileReader(
                                  source));
        while ((line=buff.readLine())!=null)
            /* stampa una riga per volta */
         {  g.drawString(line,x,y);
           if (y>=755) /*se la stampante è
                  alla penultima riga viene*/
              { /*stampato il numero della
                                    pagina
               y+=25; //spazio fra le righe
                                  stampate
               g.drawString("Pagina numero
                     "+countPages,x+100,y);
             /*stampa il num.della pag.*/
               countPages++;   /*il valore
                         del contatore  viene
                              incrementato*/
               g=pj.getGraphics();
                          /* nuova pagina */
               y=pf.getImageableX();
                  /* coordinata y per stampare
                            ad inizio foglio*/
               }
               y+=25;
           }
         y+=25;
         g.drawString("Pagina numero
                     "+countPages,x+100,y);
         buff.close();
       }catch(FileNotFoundException fnfe){}
       catch (IOException ioe) {}
       g.dispose();
       pj.end();   /*fine stampa */
    }//end stampa
}//end PrinterFast
```

Realizzare un java launcher con Dev C++

Java è un linguaggio che ha moltissimi pregi ma anche qualche piccolo inconveniente.
Un applicazione java scritta e compilata , ad esempio, sotto linux “gira” senza nessun problema sotto piattaforma windows.
La vita dello sviluppatore è stata semplificata da questa idea geniale ma da parte dell’utente è richiesto un piccolo sforzo in più.
L’utente per eseguire il nostro programmino java deve aprire la console e digitare “java ...”
Questo intervento è dovuto al fatto che i file .class o .jar non vengono considerati eseguibili.
Il programmatore allora può fornire un piccolo eseguibile che apra la console e lanci l’applicazione java.
Un programma del genere viene detto “Launcher” e si realizza in pochi minuti.
Per poter fare ciò non è necessario altro che un compilatore c++ una discreta conoscenza del linguaggio c++ e un programma java da eseguire.
Per realizzare il seguente esempio verrà utilizzato Dev-c++, un ambiente di sviluppo c++ freeware e scaricabile gratuitamente dal sito web :
www.bloodshed.net

Stefano Vena

‹1› CREARE UN NUOVO PROGETTO CONSOLE

Per prima cosa lanciamo dev-c++, poi dal sottomenu Nuovo del menu File scegliamo la voce Progetto, poi scegliamo dalla sezione basic il tipo di progetto “Windows Application” diamo un nome al nostro lavoro e diamo l’ok

‹2› SCEGLIAMO L’ICONA DEL LAUNCHER

Andiamo al menu “progetto” e scegliamo la voce “opzioni del progetto”. Ora appare il form dei settaggi selezioniamo quindi il tab “Generale”. Facciamo click sul pulsante “Libreria” per visualizzare le icone proposte da dev-c++ oppure “Sfoglia”e preleviamo un icona di nostro gradimento direttamente da hard disk

‹3› LE PRIME RIGHE DI CODICE

```
#include <stdlib.h>
#include <window.h>

int main ( int argc, char *argv[] ) {
```

Il progetto contiene già un file con un semplice un programma c++. Cancelliamo il contenuto del file main.cpp e riscriviamo daccapo il codice:
Aggiungiamo l’header <window.h> e <stdlib.h> inseriamo la segnatura del main e andiamo al passo successivo. Più facile di così…

‹4› LE VARIABILI NECESSARIE

```
DWORD size;
HKEY hKey;

char *javapos = new char[255];
bool errore = true;
```

La variabile size e la variabile hKey verrano utilizzate per accedere al registro di configurazione ed indicano rispettivamente la lunghezza del testo letto ed il puntatore alla chiave del registro che stiamo consultando. Nella variabile javapos , dopo l’interrogazione del registro di windows, verrà copiato, se esiste, il valore letto dalla chiave e precisamente il percorso in cui è stato installato jre 1.4.x. La variabile booleana errore, utilizzata come flag, viene inzializzata con il valore true. Se jre è presente sul PC dell’utente la variabile errore sarà impostata a false altrimenti manterrà il suo valore di true

‹5› CONTROLLARE L’INSTALLAZIONE DEL JRE

```
if( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
   "SOFTWARE\\JavaSoft\\Java Runtime
                      Environment\\1.4", 0 ,
KEY_ALL_ACCESS, &hKey) == ERROR_SUCCESS)
   if(RegQueryValueEx(hKey, "JavaHome", NULL ,
   NULL, (BYTE*)javapos, &size) == ERROR_SUCCESS )
     errore = false;
   RegCloseKey(hKey);
 if ( errore ) {
   MessageBox( NULL , "Java Runtime Environment
            non disponibile", "Java Launcher" ,0);
   return 0; }
```

Quando installiamo jre vengono generate una serie di chiavi nel registro di configurazione. Interroghiamo il registro di windows e se non è possibile accedere ad una chiave allora vuol dire che jre non è installato

‹6› ESEGUIRE L’APPLICAZIONE ED USCITA

```
ShellExecute(NULL, "open", "java",
           "NomeProgramma", "parametri", 1);
```

Ora siamo pronti per eseguire il nostro programma java. Inseriamo le informazioni riguardanti il file compilato all’interno del Launcher. Mettiamo al posto di NomeProgramma il nome scelto per il tuo programma java (bada bene a maiuscole e minuscole) , se sono necessari parametri aggiuntivi come librerie aggiuntive, classpath o altro, sostituiamo la stringa "parametri" con una stringa contenente queste informazioni altrimenti, se non abbiamo bisogno di altri parametri, con il valore NULL.

```
ShellExecute(NULL, "open", "java",
              "NomeProgramma", NULL 1);
```

Dal momento che in c++ non possiamo godere dei servigi del garbage collector dobbiamo liberare le risorse impiegate e poi uscire dal nostro Java Launcher personale.

```
delete [] javapos;
return 0; }
```

Creazione, lettura e scrittura di un chiave di registro di windows

Capita spesso di avere la necessità di salvare alcune informazioni relative al bel programma che stiamo realizzando.
Ma dove salvarle? La risposta è semplice: nel Registro di configurazione del windows, lontano da occhi indiscreti e in un posto dal quale non possono essere cancellate "involontariamente". Per poter fare ciò non è necessario altro che un compilatore c++ e una discreta conoscenza del linguaggio c++.
Per realizzare il seguente esempio verrà utilizzato Dev-c++, un ambiente di sviluppo c++ freeware e scaricabile gratuitamente dal sito web: *www.bloodshed.net*

Stefano Vena

<1> CREARE UN NUOVO PROGETTO CONSOLE

Per prima cosa lanciamo dev-c++, poi dal sottomenu Nuovo del menu File scegliamo la voce Progetto, poi scegliamo dalla sezione basic il tipo di progetto "Console Application" diamo un nome al nostro lavoro e diamo l'ok.

<2> LE PRIME RIGHE DI CODICE

```
#include <iostream>
#include <stdlib.h>
#include <window.h>
using namespace std;
int main ( int argc, char *argv[] ) {
```

Il progetto contiene già un file con un semplice un programma c++. Precisamente sono presenti due inclusioni *<iostream>* e *<stdlib.h>* ed una semplice implementazione di main. Aggiungiamo l'header *<window.h>* e andiamo al passo successivo.

<3> DICHIARAZIONE DELLE VARIABILI

```
HKEY hKey;
DWORD size;
char Nome_Ottenuto[100];
const char* Nome = "Nome del Programma";
size_t NomeLen = strlen(Nome);
int x = 1001;
int y = 0;
```

La variabile hKey punta ad una chiave del registro , la variabile size verrà utilizzata per memorizzare la dimensione dei dati letti dal registro nella variabile di tipo stringa Nome_Ottenuto verrà, invece, memorizzato il valore della chiave letto dal registro.
La stringa Nome_Ottenuto è stata dichiarata come array di char a dimensione fissa poiché le funzioni di lettura del registro non gestiscono le stringhe dinamiche.
Le variabili intere x e y vengono usate , prima, per impostare i valori da scrivere nelle chiavi del registro di windows, poi, come destinazione della lettura delle chiavi appena create nella fase di test.

<4> SCRITTURA DEI VALORI

```
if( RegCreateKey ( HKEY_LOCAL_MACHINE, "SOFTWARE
\\IlMioProgramma\\Finestra",&hKey) == ERROR_SUCCESS )
 {   RegSetValueEx( hKey, "Nome", 0, REG_SZ,
                          (BYTE*)Nome, NomeLen );
     RegSetValueEx( hKey, "x", 0,
                  REG_BINARY,(BYTE*) &x, sizeof(int) );
     RegSetValueEx( hKey, "y", 0,
                  REG_BINARY,(BYTE*) &y, sizeof(int) );
     RegCloseKey(hKey);  }
 else
 { cout << "Impossibile Creare la chiave di registro"
                     << endl; system("PAUSE");
   return -1;
}
```

La prima cosa da fare è creare la chiave di registro da utilizzare. Il file di registro è organizzato ad albero e ogni valore è raggiungibile attraverso un percorso. Il percorso che ci conduce alla nostra chiave è il seguente HKEY_LOCAL_MACHINE\SOFTWARE\IlMioProgramma\Finestra. Una volta creata la chiave attraverso RegCreateKey scriviamo su di essa alcuni valori conil con il metodo RegSetValueEx. Il valore REG_SZ indica che i dati che andiamo a scrivere sono stringhe, il valore REG_BINARY indica che stiamo scrivendo un generico valore binario. Se per qualche motivo la chiave non può essere creata stampiamo un messaggio di errore ed usciamo dal programma.

<5> LETTURA DEI VALORI DALLE CHIAVI

```
if( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFTWARE
   \\IlMioProgramma\\Finestra", 0 , KEY_ALL_ACCESS,
                          &hKey) == ERROR_SUCCESS)
 {  if(RegQueryValueEx( hKey, "Nome", NULL , NULL ,
    (BYTE*)Nome_Ottenuto, &size) == ERROR_SUCCESS )
          cout << "Nome =" << Nome_Ottenuto  << endl;
   if(RegQueryValueEx(hKey, "x", NULL , NULL , (BYTE*)
          &x, &size) == ERROR_SUCCESS ) cout << "
                               x ="<< x << endl;
   if(RegQueryValueEx(hKey, "y", NULL , NULL , (BYTE*)
      &y, &size) == ERROR_SUCCESS ) cout << " y ="<< y
                                      << endl;
    RegCloseKey(hKey);  }
   else
   {  cout << "Impossibile Leggere la chiave di
                        registro" << endl;   }
   system("PAUSE");     return 0; }
```

La prima cosa da fare per poter leggere i valori da una chiave e aprire la chiave stessa. Se la chiave è stata aperta con successo *RegOpenKeyEx* ritorna il valore "ERROR_SUCCESS" e la variabile hKey contiene il puntatore alla chiave richiesta. Se si verifica un errore stampiamo un messaggio di avviso per l'utente. La lettura vera e propria avviene utilizzando la funzione *RegQueryValueEx*. L'utilizzo di questa funzione è delicato in quanto richiede il passaggio dei riferimenti delle variabili x , y ed il puntatore a Nome_Ottenuto, un uso improprio potrebbe causare errori quindi è buona norma controllare sempre il valore della variabile size.

<6> IL RISULTATO

Una volta completate queste operazioni siamo pronti per vedere il codice scritto in azione! Salviamo il progetto, compiliamolo e poi lanciamo l'eseguibile creato, se tutto è andato bene allora verrà aperta la console e su di essa stampati i valori prima scritti e poi letti nel registro. Se per qualche motivo volessimo cancellare una chiave con tutte le sue sottochiavi oppure un solo attributo possiamo utilizzare la semplice funzione *RegDeleteKey(HKEY_LOCAL_MACHINE,"SOFTWARE \\IlMioProgramma\\Finestra");*

Ottenere e stampare la lista dei file di in una directory sotto windows

A volte abbiamo la necessità di dovere ottenere la lista dei file contenuti in una directory in un qualsiasi supporto di memorizzazione, per vari scopi, stamparla a schermo, contare il numero di file e di directory, cercare un particolare file ed altri motivi. Nell'esempio riportato di seguito andremo a vedere come ottenere tutti i file e le sottodirectory contenuti in un determinato percorso, utilizzando le API di Windows; Per ogni file trovato stamperemo sullo schermo il nome, qualche attributo (directory, file di sola lettura, file di sistema o file nascosto) e la dimensione in byte del file. Ciò di cui abbiamo bisogno è un compilatore c++, un editor di testo e un po' di dimestichezza con c++. Per realizzare il seguente esempio ho utilizzato Dev-C++ un ambiente di sviluppo c++ freeware. Dev-C++ è scaricabile gratuitamente dal sito web: *www.bloodshed.net*

Stefano Vena

‹1› CREARE UN NUOVO PROGETTO CONSOLE

Per prima cosa lanciamo dev-c++, poi dal sottomenu Nuovo del menu File scegliamo la voce Progetto, poi scegliamo dalla sezione basic il tipo di progetto "Console Application" diamo un nome al nostro lavoro e diamo l'ok

‹2› LE PRIME RIGHE DI CODICE

```
#include <iostream>
#include <stdlib.h>
#include <window.h>

using namespace std;

int main ( int argc, char *argv[] ) {
```

Il progetto contiene già un file con un semplice un programma c++.
Precisamente sono presenti due inclusioni *<iostream>* e *<stdlib.h>* ed una semplice implementazione di main.
Aggiungiamo l'header *<window.h>* e andiamo al passo successivo

‹3› DICHIARAZIONE DELLE VARIABILI

```
WIN32_FIND_DATA FindFileData;
 char directory = "C:\\*.*" ;
 bool isDir = false;
```

Le variabili necessarie sono tre: la stringa che contiene il percorso della directory da scandire, un valore booleano, utilizzato come flag e indica se un elemento selezionato durante la scansione della directory è a sua volta una directory oppure un file. La terza variabile utilizzata è un struttura nativa del windows (WIN32_FIND_DATA) che contiene delle informazioni sui file e verrà aggiornata ad ogni passo della scansione. E' necessario che la stringa con il percorso di ricerca finisca con il nome di un file o con una porzione di esso combinato con i caratteri jolly * oppure ? così come avveniva nella sintassi di utilizzo del buon vecchio comando dir del DOS. Ricorda che il carattere '\' in c++ è un carattere di comando quindi è necessario utilizzare la sequenza '\\' per identificare il carattere back slash

‹4› LA PRIMA OCCORRENZA

```
HANDLE hFindFile = FindFirstFile( directory,
                              &FindFileData );
BOOL continua = ( hFindFile !=
                          INVALID_HANDLE_VALUE );

while( continua )
{
```

La prima funzione che utilizziamo è FindFirstFile, essa prende come argomenti la stringa directory ed il riferimento alla struttura FindFileData dichiarata in precedenza,il valore ritornato è un handle. Se tale valore non è valido, cioè è pari a INVALID_HANDLE_VALUE, alla variabile booleana continua verrà assegnato il valore FALSE e il ciclo di while utilizzato per la ricerca dei file non verrà avviato. Questo evento si verifica se la directory risulta vuota o se il percorso è inesistente!

‹5› STAMPA DELLE OCCORRENZE

```
 size_t size = (FindFileData.nFileSizeHigh *
                                  MAXDWORD) +
FindFileData.nFileSizeLow;
 isDir = (FindFileData.dwFileAttributes &
                    FILE_ATTRIBUTE_DIRECTORY);

 printf("  %-6s %9d byte %s\n",
      (isDir ? "<DIR>" : "<FILE>") ,
  size ,
  FindFileData.cFileName);
```

Ad ogni occorrenza ricaviamo alcune informazioni utili e poi le stampiamo in uscita.
Ricaviamo la dimensione in byte del file tramite una formulina e assegnamola alla variabile size, controlliamo se l'elemento corrente e un file oppure una directory ed infine stampiamo a schermo le informazioni ottenute con la funzione printf ne seguente formato <TIPO> (%-6s) dimensione (%9d byte) Nome dell'elemento (%s).
Un esempio è il seguente:

```
<DIR >   0 byte WINDOWS
 <FILE>   5 byte CONFIG.SYS
```

‹6› IL GRAN FINALE

```
  continua =  FindNextFile( hFindFile , &FindFileData );
 }

 FindClose( hFindFile );

 system("PAUSE");
 return 0;
}
```

Dopo avere stampato le informazioni relative al file corrente procediamo con la ricerca del file successivo. Se esiste un file successivo la variabile continua riceverà il valore true che fa continuare il ciclo di while altrimenti il ciclo si interrompe. Una volta usciti dal ciclo rilasciamo le risorse associate allo handle utilizzato. L'utilizzo della funzione system("PAUSE") è un espediente per evitare che il programma chiuda la console di DOS prima di aver ammirato il risultato.

Edisplay Start Il tuo negozio OnLine

Questo mese con ioProgrammo trovate un regalo eccezionale: il software Edisplay in versione Start, la cui versione commerciale ha un valore di € 120

Edisplay è una suite per la generazione di negozi basati su sistemi di commercio elettronico. Detto in parole molto povere, potete creare un sito di commercio elettronico completo veramente con pochissimi clic e nonostante questo offrire tutte le funzioni indispensabili per un buon sito ecommerce elettronico.

UN SISTEMA INTELLIGENTE

La prima cosa interessante da mettere in evidenza in Edisplay è la modalità con cui è stato concepito il sistema. C'è una netta divisione fra l'interfaccia OnLine disponibile al cliente e l'interfaccia di gestione del negozio disponibile al negoziante. La prima è un normale insieme di pagine html e script asp o perl, la seconda è invece un'applicazione standalone che gira fisicamente su un computer locale anche non connesso a internet in modo permanente. L'interfaccia *"OnLine"* viene "creata" a partire da quella OffLine. Il negoziante effettua tutte le modifiche importanti al catalogo, ai prodotti, ai prezzi, alle modalità di presentazione, effettua poi l'operazione di "generazione" del negozio. Il risultato è un insieme di file html, asp o perl che viene inviato via FTP all'host che fisicamente contiene il negozio.

Fig. 1: ***Edisplay offre un ottimo supporto anche online***

I VANTAGGI DI UNA GENERAZIONE ASINCRONA

Ci sono alcune considerazioni per cui questo sistema ci è sembrato interessante. Prima di tutto bisogna pensare che il prodotto almeno nella versione Start è pensato per piccoli negozi online, che normalmente non dispongono di personale qualificato per la gestione si sistemi complessi, per giunta via web. L'interfaccia standalone è un buon modo di semplificare la vita al gestore del negozio. Il secondo vantaggio è rappresentato dalle modalità di generazione del negozio. Se avrete la pazienza di andare solo poco più in profondità, basterà uno sguardo per capire che quasi tutto quello che viene generato è costituito da normali file *.html*, e pochissimi script di contorno e molto ben isolati dal resto dell'applicazione. Questo significa che chiunque conosca un minimo di Html è in grado di personalizzare l'aspetto del negozio, di modificare i file, avendo semplicemente l'accortezza di non toccare gli script e i link che li richiamano. Le modifiche si possono facilmente fare anche con Dreamweaver

CARATTERISTICHE DEL PRODOTTO

Il negozio ha le caratteristiche tipiche di tutti i sistemi di ecommerce esistenti. Gestisce il carrello, gli ordini, la divisione in categorie, il pagamento elettronico, la ricerca facilitata, le offerte. Quello che è interessante è la pipeline che conduce all'ordine. Il sistema è stato studiato anche questa volta per portare l'uten-

COME INIZIARE

1) **Avvia l'installazione dal file setup_start_em.exe**
2) **Al termine dell'installazione avvia l'interfaccia di gestione tramite l'icona che comparirà sul tuo desktop**
3) **Effettua la registrazione gratuita del prodotto all'indirizzo: http://www.edisplay.it/web/start_register.php**
4) **Ricopia il numero seriale che ti sarà inviato via email nell'apposita finestra di dialogo che comparirà quando avvierai per la prima volta il programma**
5) **Segui il nostro tutorial a fondo pagina per generare il vostro primo negozio online**

1 DEFINISCI I REPARTI - In questa sezione è possibile definire le categorie a cui i singoli articoli devono appartenere. È come individuare i reparti di un negozio

2 DEFINISCI GLI ARTICOLI - Inserisci gli articoli nel catalogo e stabilisci le caratteristiche che li definiscono, prezzo, peso immagine

3 LE CARATTERISTICHE DEL NEGOZIO - Stabilisci le caratteristiche generali del sito in termini di valuta, possibilità di ricerca, visualizzazione

4 DEFINISCI I SISTEMI DI PAGAMENTO - Scegli fra i vari sistemi di pagamento quello che è attinente con le scelte commerciali che hai fatto

5 SETTA I PARAMETRI PER IL SERVER FTP - Definisci i parametri del server che ospiterà il sito. In particolare user e password per le varie cartelle

6 SCEGLI IL TEMPLATE - Scegli l'aspetto estetico da uno dei template disponibili. Potrai comunque modificarlo in seguito senza troppi problemi

7 COSTRUISCI IL NEGOZIO - Dai il via alla fase di costruzione del negozio. In questa fase verranno generati i file che compongono il negozio

8 EFFETTUA L'UPLOAD - Trasferisci i file sul sito online. Una volta compiuta questa operazione avrai terminato! Non ti resta che aspettare i clienti

te all'acquisto in appena tre clic. Non è necessaria nessuna registrazione, anche se gli utenti che volessero potranno salvare i propri dati per riutilizzarli in seguito, è prevista l'integrazione con i maggiori circuiti di pagamento. Quello che è più sorprendente e che poco o niente dovrà fare il negoziante dal punto di vista tecnico per integrare i sistemi di pagamento elettronici nel proprio negozio. Utilizzando paypal ad esempio il tutto si riduce a un paio di clic nell'interfaccia di gestione.

I SISTEMI OPERATIVI SUPPORTATI

L'interfaccia di gestione è attualmente un'applicazione Windows Standalone. Dal punto di vista server invece è possibile scegliere fra sistemi Windows e sistemi Unix. Nel primo caso gli script che gestiscono il negozio saranno generati in linguaggio ASP, nel secondo caso verranno generati degli script in perl. Anche questo è un grande vantaggio per chi decide di utilizzare il prodotto. Edisplay può naturalmente essere utilizzato all'interno di una propria area di gestione, ma sarà più frequente proiettarlo in sistemi in Hosting presso un qualche provider. L'opportunità di poter scegliere la piattaforma su cui il negozio dovrà girare, svincola ulteriormente il commerciante dal doversi interessare a problematiche di ottimizzazione, il negozio funziona bene su tutte le piattaforme, pertanto il negoziante potrà concentrarsi unicamente su quello che gli compete, ovvero la parte prettamente commerciale e valutazione del reddito.

Solo per i lettori di ioProgrammo
Puoi acquistare le versioni Base e Pro di Edisplay con uno sconto eccezionale. L'offerta è disponibile all'indirizzo:
http://www.edisplay.it/riviste_2005/cat/

Fig. 2: Un sito demo realizzato con Edisplay

SOFTWARE SUL CD

MediaWiki 1.3.11

La vostra wikipedia personalizzata

Un wiki è sito internet gestito da qualcosa di molto simile a un CMS. Così come un CMS viene utilizzato per l'inserimento rapido di contenuti sul Web, a differenza di un CMS però dispone di una sorta di linguaggio interno tale che se all'interno del testo vengono trovati dei marcatori particolari automaticamente viene creata una pagina web vuota che li referenzia.

Questo sistema garantisce la creazione rapida di enciclopedie che sfruttano al massimo il sistema di HyperLink tipico di internet. Inoltre le pagine "vuote" che vengono create automaticamente dal sistema possono essere riempite dagli utenti che ne hanno i permessi e così via con un meccanismo piramidale. Comprenderete che creare un'enciclopedia utilizzando questo sistema è piuttosto semplice. MediaWiki è il software che serve una delle più grandi enciclopedia mondiali online: WikiPedia - *http://it.wikipedia.org*.

Directory: / WikiMedia

EZstream 0.2.0

L'encoder per ICECast

Si tratta di una piccola utility a linea di commando, gestita da un file XML che consente di fare l'encoding della vostra musica e redirezionarla su un server ICECast. Esiste anche qualcosa di più complesso e graficamente accattivante per gestire questo genere di servizio, tuttavia proprio per la sua leggerezza EZStream rappresenta un'ottima soluzione.

Directory: /EZstream

Graphics3D C++ 6.05

Una libreria ad alte prestazioni per la grafica

Dedicata non solo agli sviluppatori ma anche a ricercatori e studenti; supporta diversi sistemi operativi e linguaggi (Windows MSVC++ 6, Windows Visual Studio .NET (MSVC++ 7.0), Linux x86 gcc 3.3, e OS X Xcode) Incorpora funzioni per la programmazione dell'hardware GLSL, OpenGL ARB assembly, NVIDIA assembly, e specifiche Cg.

Directory: /G3d

Agast 1.1

Chi non ha mai provato a sviluppare un videogioco scagli la prima pietra. Con Agast questo non sarà più un problema.

Non sempre rapido vuol dire anche potente e flessibile. In questo caso non è proprio così. Agast è un tool rapido per lo sviluppo di videogame in stile Monkey Island. È semplicissimo da usare, ma anche molto potente. Assolutamente da provare.

Directory: /Agast

Filezilla

I sorgenti di uno dei client FTP OpenSource più usati

Se avete bisogno di inserire un client FTP in una vostra applicazione, studiare come sono composti questi sorgenti può essere un ottimo inizio. FileZilla include moltissime funzionalità tipiche di un client FTP evoluto. Se volete personalizzare il software esistente oppure riprodurre queste caratteristiche senza dubbio questi sorgenti vi saranno d'aiuto.

Directory: /filezilla-src

Mambo V4.51

Il CMS professionale

Di sistemi di CMS, soprattutto OpenSource ne esistono un quantitativo strabiliante. Molti sono fork del capostipite PHPNuke, molti sono progetti nati da singoli sviluppatori, pochi sono così professionali e completi come Mambo.

Questo CMS fa della modularità e della sua capacità di essere facilmente e completamente personalizzabile il suo punto di forza. Inoltre esistono centinaia di moduli per Mambo, e se avrete voglia di sviluppare qualche estensione per questo CMS, scoprirete anche che la sua architettura è tale che sviluppare un modulo personalizzato è sufficientemente semplice.

Directory :/Mambo

Wordpress 1.5

Il blog per eccellenza

Se Drupal è un Blog Container, ed EZPublish un framework per la costruzione di CMS, Wordpress è un software per la costruzione di un solo singolo Blog.

Utilissimo per la creazione di un piccolo blog personale che potrete gestire in modo molto elementare.

Tutta questa semplicità non vi inganni, Wordpress è il Blog più usato al momento su internet e dispone di una tale quantità di moduli aggiuntivi che potrete, se volete, personalizzarlo come meglio credete.

Directory: /Wordpress

Ezpublish 3.5.1

Il framework per la costruzione di CMS

Sebbene ad un primo sguardo, dopo l'installazione, potreste pensare di trovarvi davanti ad un normale sistema CMS, EZPublish è molto di più. Si tratta infatti di un completo framework RAD per la costruzione di applicazioni di gestione contenuti molto specializzate.

È possibile definire nuove classi, nuovi formati per l'input, definire il workflow dell'applicazione, elaborare template molto accurati utilizzando Smarty.

Si tratta insomma di un prodotto molto evoluto con potenzialità straordinarie.

Directory: / Ezpublish

Icecast 2.2.0

Il server di streamer OpenSource

Avete clienti che vi hanno chiesto di portare la propria radio su internet? Più semplicemente volete provare a mettere in piedi una radio che trasmetta solo su internet? Icecast, almeno dal punto di vista software, è quello che fa per voi!. Avete bisogno di un encoder che prenda il suono dalla scheda audio, lo trasmetta al server Icecast e il gioco è fatto. Qualunque client connettendosi al server Icecast potrà ascoltare la vostra musica.

Directory: /Icecast

1 INSTALLA E CONFIGURA ICECAST - L'installazione di IceCast è banale, e rispecchia il classico wizard di setup di tutti i programmi windows. Una volta installato, dopo avere lanciato l'applicazione si può accedere al file di configurazione tramite il menu "Edit"

```
<authentication>
    <source-password>hackme</source-password>
    <relay-password>hackme</relay-password>
    <admin-user>admin</admin-user>
    <admin-password>hackme</admin-password>
  </authentication>
<listen-socket>
    <port>8000</port>
  </listen-socket>
```

2 MODIFICA LA CONFIGURAZIONE - Se non hai esigenze particolari, non c'è motivo di cambiare la configurazione iniziale. Potrai studiare con calma tutte le opzioni in un secondo momento. Prendi solo nota della porta dove girerà il server: la 8000 in questo caso

3 INSTALLA EZSTREAM - Ezstream è il nostro encoder, prenderà i file .mp3 da una directory locale sul computer o catturerà il suono dalla scheda audio e lo invierà al server Icecast pronto per essere diffuso su internet. Puoi usare l'Eucoder che preferisci.

4 CREA UNA PLAYLIST - Puoi usare ad esempio Winamp - http://www.winamp.com per creare una playlist e salvarla come playlist.m3u nella stessa directory dove hai installato ezstream. Una playlist è un elenco di file mp3 che vuoi mandare in streamimg

5 AVVIA IL SERVER ED EZSTREAM - Lancia prima il server, utilizzando il bottoncino start nell'interfaccia di amministrazione di Icecast. Poi lancia una console dos e portati nella directory di installazione di ezstream. Lancia l'encoder con ezstream -c ezstream_mp3.xml

6 INVITA GLI AMICI - A questo punto chiunque possieda un client e conosca il tuoi indirizzo IP può collegarsi alla tua radio. Ad esempio usando winamp è sufficiente utilizzare CTRL+L e inserire http://tuoindirizzoip:8000/stream e sarai in onda!

MySQL Administrator

Il front-end per amministrare mysql in modo grafico

La grande diffusione di MySql degli ultimi tempi è avvenuta anche grazie al proliferare di interfacce grafiche che aiutano l'utente nella sua amministrazione ordinaria. È il caso di MySQL Administrator che fornisce all'utente un potente, completo quanto comodo sistema per la gestione di ogni aspetto di mySQL. Si parte dalla creazione degli utenti fino alla gestione dei permessi, alla creazione dei database, alle statistiche sull'uso del server.

Directory: /mysql-administrator-1.0.19

EclipseME 0.7.5

Lo sviluppo mobile secondo Eclipse

EclipseME è un plug-in per Eclipse che vi aiuta nella creazione di una MIDlet Suite (un pacchetto software che può includere varie applicazioni e file di risorsa) e dei MIDlet veri e propri (le applicazioni J2ME).

Directory: /EclipseME

Drupal 4.5.2

Il costruttore di Blog

Drupal è prima di tutto un sistema di CMS, consente cioè la pubblicazione dei contenuti in modo semplificato, di modo che anche gli utenti meno smaliziati possano inserire dei contenuti su internet pur non avendo conoscenze di programmazione o di html, ma drupal è anche molto di più. Utilizzando Drupal ognuno degli utenti iscritti al sito diviene proprietario di un blog, e con pochi accorgimenti potrete fare in modo che ciascuno di questi blog goda di una vita propria indipendente dal sito principale, oppure se lo desiderate potrete far comunicare i vari blog in modo da creare una vera community di bloggers.

Directory: /Drupal4.5.2

1 INSTALLARE I FILE - Prendi il file drupal-4.5.2.tar.gz e scompattalo in una directory visibile sul webserver. Puoi usare un qualunque programa di decompressione per fare questo, ad esempio winrar. È utile rinominare la directory in questione con un nome facile, ad esempio "Drupal".

```
mysqladmin -uroot -platuapassword
                        create drupal
mysql -uroot -latuapassword
mysql >> GRANT ALL PRIVILEGES ON
   drupal.* TO drup_user@localhost IDEN-
          TIFIED BY 'passwordutente';
mysql >> exit
cd [directory di installazione di drupal]
mysql -udrup_user -ppasswordutente
drupal < database/database.mysql
```

2 INSTALLA IL DATABASE - Apri una console msdos e segui le istruzioni contenute nel riquadro, servono rispettivamente per creare un database, creare un utente con password per il database in questione, riempirlo con dei dati che servono a far funzionare drupal

3 AVVIA DRUPAL - Punta il browser all'indirizzo http://localhost/drupal. Se tutto è andato a buon fine vedrai la schermata iniziale di drupal. Se vedi dei warning forse stai usando PHP5, drupal è testato solo per PHP4, ha qualche problema con la versione 5. Controlla il php.ini

4 CREA UN UTENTE - Clicca su "*Create new Account*" e segui le istruzioni per creare il primo utente. Il primo utente creato è anche l'amministratore dell'intero sito. Gode di tutti i privilegi per eseguire i vari settaggi e configurare drupal in modo opportuno

5 QUALCHE CONFIGURAZIONE - Loggati usando il tuo nuovo utente amministrativo e clicca sinistra su "Administer" e poi su "Settings", da qui puoi iniziare a configurare le cose essenziali, ad esempio il nome del sito, lo slogan, eventuali permessi

6 SCRIVI IL TUO PRIMO ARTICOLO - Clicca su "*Create Content*" e poi su "*story*", ti si aprirà una pagina in cui potrai scrivere il tuo primo articolo, inoltre potrai decidere se postare l'articolo in home page oppure semplicemente lasciarlo a disposizione

PHPbb 2.0.13

Il forum più diffuso sul Web

Se volete dotare il vostro sito Web di un Forum, PHPbb è quello che fa per voi. È stato per lungo tempo il dominatore incontrastato dei forum open-source disponibili su internet, ha subito recentemente qualche critica dovuta a problemi di sicurezza, eppure è ancora uno dei sistemi più usati su Internet. Basato sull'ormai arcinota accoppiata PHP+MySQL rappresenta senza dubbio un'ottima soluzione per i vostri Bulletin Board.

Directory: /PHPbb

```
mysql -uroot -plavostrapassord
mysql> create database phpbb_db;
mysql> grant all privileges on phpbb_db.
        * to phpbb_user@localhost identified
                              by 'unapassword';
mysql> exit;
```

1 PRIMA DI INIZIARE - Scompattate il file phpBB-2.0.13.zip all'interno di una directory visibile sul vostro webserver. Abbiate cura di usare un nome semplice ad esempio "Forum". Aprite una console dos e seguite i passi nel riquadro per creare il database.

2 L'INSTALLAZIONE - Puntate il browser su http://localhost /path_di_phpbb. Si aprirà una schermata contenente una serie di campi da riempire per dare il via all'installazione. Usate le password e i nomi di database che avete utilizzato al passo uno.

3 LA ACCENDIAMO? - Confermate che l'installazione è andata a buon fine cancellando le directory install e contribute. Eseguite poi un refresh del browser tramite il tasto F5. Vi comparrà una finestra di login, loggatevi con il nome dell'amministratore

4 AMMINISTRIAMO - In basso nella pagina cliccate su "Go to Administration Panel". Da questo link si accede a tutte le opzioni di amministrazione. Potrete creare nuovi forum, amministrare gli utenti, gestire gli argomenti, e molto altro ancora

5 CREIAMO UN FORUM - A sinistra nella colonna verticale clicchiamo su "Management" sotto la tabsheet "Forum Admin". Riempiamo i campi che compaiono nella parte destra della pagina con i dati necessari alla creazione del forum, confermiamo il tutto

6 DIAMO UNO SGUARDO - Clicchiamo in alto su Preview Forum, e se tutto è andato a buon fine torneremo alla pagina principale del forum, che questa volta contiene anche una discussione che ha per argomento il nuovo forum appena creato

Blender 2.36

Un modeller per oggetti 3D

Blender non è propriamente un tool riservato ai soli programmatori, piuttosto può aiutare alcune categorie di programmatori nello sviluppo dei propri software. Si tratta infatti di un ambiente per lo sviluppo di modelli tridimensionali. Indispensabile per chi per esempio sviluppa videogame gode di alcune importanti proprietà.
È un prodotto GPL perciò privo di costi diretti, è straordinariamente potente e completo di tutte le caratteristiche dei più noti software commerciali, è in grado di esportare i dati in formati comprensibili dalla maggior parte degli engine 3d ivi compresi irrlicht e directx.

Perciò se siete degli sviluppatori di videogame o se in un qualche modo sviluppate dei prodotti attinenti al mondo tidimensionale, Blender senza dubbio vi può essere utile.

Directory: /Blender3D

Irrlicht 0.8

Accendi il miglior motore 3D open source!

IrrLicht è un motore per la grafica tridimensionale, scritto in C++ e utilizzabile sia con questo linguaggio, sia con i linguaggi di .NET.
Presenta le principali caratteristiche

che si trovano anche nei motori professionali e vanta una notevole comunità di sviluppatori, con diversi progetti in attivo.

IrrLicht ha tra i suoi pregi anche quello di poter utilizzare come librerie

Directory: /Irrlicht

JBOSS 4.0.1

L'application Server per applicazioni J2EE

Qualcuno di voi avrà già sentito parlare di J2EE. Si tratta delle specifiche Sun per costruire applicazioni distribuite adatte ad un ambiente di business.

Per applicazione distribuita si intende un'applicazione tale che le sue varie parti non risiedono su un unico computer ma vengono invece reperite su macchine e sistemi differenti e forniscono servizi e informazioni all'applicazione che le utilizza. Allo stesso modo qualcuno avrà sentito la denominazione di Application Server. Un'Application Server è un contenitore di servizi che vengono esposti e resi disponibili alle applicazioni che ne fanno richiesta. JBoss AS è un Application Server conforme alle specifiche J2EE, è cioè un contenitore di servizi basati su applicazioni Java e resi disponibili in un ambiente distribuito. In questo numero di ioProgrammo potete leggere il bell'articolo di Ivan Venuti per saperne di più.

Directory: /jboss401

Struts 1.2.4

Il framework per costruire applicazioni JSP conformi al pattern MVC

Di Struts abbiamo parlato molto parlato nei numeri di ioProgrammo precedenti, si tratta di un framework per costruire applicazioni Web indistruttibili usando Java. La sua solidità è dovuta al fatto di fare riferimento al pattern MVC, che è stato appositamente studiato per realizzare applicazioni facilmente mantenibili e incredibilmente solide.

Directory: /Struts

Spe 0.7.0

Un editor evoluto per Python

La popolarità di un linguaggio si evince anche dal numero di tool e di librerie che vengono prodotte a suo supporto. Se dal punto di viste delle librerie Python può vantare un conspicuo numero di prodotti che ne estendono le caratteristiche, dal punto di vista invece dei tool ancora non avevamo visto degli editor sufficientemente evoluti da poter supportare pienamente lo sviluppo con Python. Spe colma questa lacuna. Si tratta di un editor decisamente evoluto completo di code completion e di sintax highlighting oltre che di tutte le caratteristiche tipiche di un editor professionale. Sia che stiate imparando a programmare in Python sia che siate dei programmatori abituali, sicuramente Spe rappresenta un'ottima scelta.

Directory: /Spe

FreeTTS 1.1.2

Un sintetizzatore vocale scritto interamente in Java

A freeTTS abbiamo dedicato ampio spazio nei numeri precedenti di ioProgrammo. Si tratta di un motore sintesi vocale interamente scritto in Java. Dove per sintesi vocale si intende che questo engine consente di creare applicazioni Java in grado di produrre dei suoni molto simili alla voce umana. Che ne dite ad esempio di farvi leggere una mail dalla vostra applicazione Java?

Directory: /FreeTTS

Sphinx4

Un riconoscitore di parole

Se freeTTS è un sintetizzatore vocale, mette cioè in grado le vostre applicazioni di parlare, Sphinx4 è il suo opposto corrispondente, consente cioè di scrivere applicazioni in grado di riconoscere il suono della parola umana e compiere delle azioni specifiche in corrispondenza di determinate parole. Pensate ad un Call center automatizzato, Sphinx4 consente di creare applicazioni di questo tipo. Ne abbiamo a lungo parlato nei numeri precedenti di ioProgrammo

Directory: /Sphinx4

Simple Direct Media Layer

Una libreria per la gestione del Multimedia

Ce ne parla a lungo Daniele De Michelis in un bell'articolo pubblicato in questo stesso numero di ioProgrammo. Si tratta di una libreria cross platform in grado di gestire a basso livello la tastiere, il mouse, le funzioni 3D della scheda video, eventuali joystick è molto altro ancora. Assolutamente da provare se volete spingervi nella grafica avanzata.

Directory: /SDL

Mrtg 2.11.1

Misurazioni di rete perfette

Volete sapere quanto consuma in termini di banda il vostro WebServer? Avete necessità di conoscere quanta banda viene consumata in termini di accessi FTP? Mrtg è un frontend verso il servizio SMTP . Consente di misurare con estrema precisione tutti i parametri che caratterizzano il vostro sistema sia esso un sistema Window che Linux. Requisiti fondamentali per utilizzare a fondo questo frontend sono un'installazione di Perl funzionante sulla propria macchina e che il servizio Snmp sia attivo.

Directory: /MRTG

INDISPENSABILI

Apache 2.0.53 NEW

Uno dei server Web più usati al mondo

Un'indispensabile ormai. ioProgrammo lo ripropone spessissimo fornen-

dovi sempre le versioni più aggiornate. In questo numero Apache sarà usato per gestire più di un progetto. Si va da EZPublish a Tomcat, e chiaramente all'integrazione con PHP. Se avete bisogno di un server Web per provare le vostre applicazioni Web, oppure da usare in sistemi di produzione, Apache è quello che fa per voi.
Directory: /Apache/

Tomcat 5.5.7 NEW
Il servlet container per Java e JSP
L'idea è molto semplice. Sviluppare in java pagine Web. Ad un primo sguardo, Tomcat, potrebbe sembrare un normale WebServer. Ed in effetti è un normale WebServer! In grado di soddisfare le richieste per qualunque pagina Html. In realtà però Tomcat è anche qualcosa in più. Di fatto però Tomcat offre qualcosa in più, ovvero la capacità di soddisfare richieste per applicazioni Java. Potrebbe sembrare complesso, in realtà lo è meno di quanto sembri. Immaginate Tomcat come un grande contenitore al cui interno ci sono altri contenitori ciascuno dei quali rappresenta un'applicazione Java, che richiamata da luogo ad una pagina HTML interpretabile da un browser. Questo consente di sviluppare pagine Web (JSP) utilizzando tutta la potenza della normale gerarchia di classi Java e la sintassi e il linguaggio che qualunque programmatore Java conosce bene.
Directory: /tomcat

Dev C++ 5 Beta 9
Un editor C++ a basso costo
Dev C++ è un editor distribuito su licenza GPL, come tale non ha costi relativi al diritto d'autore. Come tutto o quasi tutto il software GPL la sua economicità non è affatto sinonimo di scarsa qualità. Al contrartio Dev C++ è uno degli editori più amati ed utilizzati da chi sviluppa in C++. Le caratteristiche sono notevoli. Si va dal Debugger integrato, al project Manager, al Class Browser, al Code Completion.
L'insieme di queste caratteristiche, oltre una leggerezza innata dell'ambiente lo rende particolarmente comodo da utilizzare per sviluppare progetti C++ anche di grandi dimensioni
Directory: /Devc++

Dev-PHP 2.0.9
Ottimo editor PHP OpenSource
Se state iniziando a sviluppare in PHP avrete bisogno di un editor. Scrivere codice con il notepad può essere un esercizio divertente, ma quando iniziate a scrivere script leggermente più complessi si impone la scelta di passare a un editor più completo. DEV-PHP non solo è completo ma anche molto potente. Dotato di code completion, sintax highlighting, funzionalità di ricerca avanzate ed una serie di tool piuttosto interessanti rappresenta una grande scelta per programmare in PHP. Inoltre è un editor straordinariamente leggero, oltre che gratuito ed OpenSource. Da non perdere!
Directory: /DevPHP

Lazarus 0.9.4
Un ambiente RAD piu copilatore Freeware per object Pascal
Volevate imparare a programmare in Delphi ma non avevate la possibilità di comperare il costoso ambiente di casa Borland?

Lazarus è ciò che fa per voi. Si tratta di un clone Freeware del noto Delphi. La somiglianza è incredibile! Il modo di procedere altrettanto! Lazarus è un RAD funziona con la stessa logica di Delphi, supporta la VCL, è dotato di tutti i componenti classici di Delphi, è sufficientemente veloce e affidabile. Certo, non è il Borland Delphi 2005 con tutte le sue infinite possibilità, ma rappresenta un'ottima base per chi vuole sviluppare applicazioni Standalone utilizzando la produttività classica di un ambiente RAD come quelli famosi di casa Borland. Si tratta di un tool eccezionale che non mancherete di iniziare ad apprezzare per tutte le sue caratteristiche.
Directory /Lazarus

Eclipse 3.0.1
La piattaforma universale multifunzione
Sembra un sottotitolo esagerato ed eclatante: "La piattaforma universale multifunzione" ed invece Eclipse è nato proprio con questo scopo. A prima vista sembra un normale editor per programmatori Java, anzi molto di più che un normale editor, visto che ospita una serie di funzionalità piuttosto avanzate che vanno dal code completion alla syntax highlithing al refactoring e che lo stanno portando a diventare uno standard proprio per Java, tuttavia è anche vero che Eclipse è completamente estensibile per mezzo di plugin, tanto che può essere utilizzato da programmatori PHP come da programmatori C++ e persino come frontend verso database etc. Insomma qualunque tipo di necessità voi abbiate, Eclipse è in grado di aiutarvi. Se poi siete dei programmatori Java rientra in quel ristretto numero di software indispensabili per gestire rapidamente il vostro lavoro.
Directory: /Eclipse

Python 2.4
Un linguaggio orientato agli oggetti con tanto di supporto a classi ed ereditarietà
Viene usato in una varietà di applicazioni. A quanto pare è largamente utizzato ad esempio da Google per lo sviluppo delle loro applicazioni.
Si caratterizza per la gestione dinamica della memoria, per l'elevata portabilità, per la curva di apprendimento relativamente breve. Un linguaggio di programmazione di cui sentiremo parlare a lungo.
Directory: /Python

j2sdk 1.5.0.01
L'indispensabile ambiente di sviluppo SUN
Se siete sviluppatori Java o avete intenzione di imparare a programmare in Java avete sicuramente bisogno del JDK. In questo numero vi presentiamo la versione 1.5.0, rilasciata già da qualche mese e di cui ioProgram-

mo si sta occupando ampiamente in ogni numero. Per utilizzare il J2SE è convieniente utilizzare un editor. Il più usato e anche forse il migliore è Eclipse di cui trovate una versione in questo stesso numero. L'accoppiata Eclipse più Java è di sicuro successo. Le caratteristiche di Eclipse sono interessanti. Si va dal code completion alla sintassi highliting al refactoring. Sicuramente è un ottimo editor alternativo ad Eclipse ed altamente estendibile grazie ai moduli.

Directory: /jdk1.5.0

PHP 5.0.3

Il linguaggio di scripting per il web

Ormai PHP lo conoscete tutti, e se non lo avete mai usato, sicuramente vi sarà capitato di accedere a qualche sito web sviluppato con PHP. Saprete dunque perciò che è un linguaggio di scripting particolarmente utilizzato per sviluppare Web Application.

Incredibilmente potente, fa della completezza del linguaggio, della facilità di apprendimento, della capacità di integrarsi con applicazioni di Dababase i suoi punti di forza.

Directory: /PHP

Sharp Develop 1.0.3

Un ambiente di sviluppo particolarmente interessante sia per .NET che per Mono

Chi pensa che Visual Studio o Web Matrix siano le uniche possibilità per programmare in .NET si sbaglia.

Esiste anche questo interessantissimo Sharp Develop che oltre ad essere un IDE fatto straordinariamente bene, ha il grosso vantaggio di funzionare sia in ambiente Windows su Microsoft .NET che in ambiente Linux su Mono.

Directory: /sharpdevelop

Ultimate++

Un fantastico ide "quasi RAD" per le applicazioni C++

Si tratta di un ambiente di sviluppo per applicazioni C++, nella versione allegata a ioProgrammo lo trovate abbinato al compilatore Mingw, nonostante questo può essere utilizzato anche con altri compilatori.

L'ambiente offre tutte le caratteristiche classiche di un ambiente professionale, parliamo di code completion, debugging, sintax highlighting.

La caratteristica più interessante dell'IDE è che è dotato di un minimo di funzionalità RAD il che dato i bassi costi del pacchetto lo rende particolarmente attraente agli occhi degli sviluppatori.

Directory: /upp-minggw-0.98.4a

MySQL Query Browser

L'editor SQL per MySQL

Anche in questo caso si tratta di un editor visuale. Attenzione non RAD ma semplicemente visuale. Le query vengono costruite in parte in modo manuale in parte è possibile utilizzare dei selettori sotto forma di bottoni che aiutano a sviluppare query sintaticamente corrette. MySQL Query Browser è un tool che può aiutare in tutte quelle situazioni dove è necessario costruire query complesse che devono essere salvate, riviste, rieditate più di una volta prima di arrivare a un risultato soddisfacente.

Directory: /mysql-query-browser

MySQL 4.1.10

Il server di database Open Source più diffuso al mondo

MySQL è un'indispensabile. ioProgrammo ogni mese vi offre la versione aggiornata. Si tratta del server di database fondamentale per la maggior parte delle applicazioni Internet scritte in PHP. Ma è difussissimo anche per le applicazioni Standalone e grazie ai nuovi connector comincia a essere usato anche dagli sviluppatori .NET.

Directory /Mysql

HSqldb 1.7.3

Un database piuttosto potente

Nato in sordina come prodotto OpenSource, HSqlDB in breve tempo sta conquistando gli onori della cronaca grazie alla sua eccezionale velocità, semplicità d'uso e affidabilità. Si tratta di un Database completamente scritto in Java che occupa appena 100k ma che espone caratteristiche interessanti. Sembrerebbe che proprio HSqlDb sia stato scelto come prodotto di base per lo sviluppo del nuovo applicativo di database che completerà la suite di OpenOffice nella sua versione 2.0. Questo testimonia anche la validità del prodotto, se ce ne fosse ulteriore bisogno.

Directory /hsqldb

Webalizer 2.0.1

Statistiche sempre aggiornate per il tuo sito Web

Webalizer è da considerare un precursore dei tempi. Si può dire che webalizer è nato insieme ai primi siti commerciali su Internet. Si tratta di un analizzatore di Log, tale che sulla base dei log analizzati produce statistiche piuttosto accurate sul sito web che ha prodotto i log.

È compatibile sia con IIS che con Linux, e proprio grazie al fatto che le statistiche vengono elaborate su un'analisi diretta dei Log è in grado di produrre statistiche molto accurate. C'è da dire che proprio per le sue potenzialità, semplicità d'utilizzo e di installazione, Webalizer è diventato

nel tempo il sistema di statistiche più utilizzato in ambiente di hosting e tutto questo nonostante il codice non venga più aggiornato da molto tempo
Directory: webalizer2.01

Xamp-1.4.12

Installare PHP+Apache+MySQL in un sol colpo

Quante volte avete provato a imparare PHP e vi siete fermati davanti alla difficoltà di installazione e configurazione del sistema? Essere un ottimo programmatore non sempre corrisponde a essere anche un ottimo sistemista. Xamp vi mette a disposizione un sistema rapido per l'installazione di tutto l'occorrente per lavorare con PHP. Installa Apache come server Web, MySQL come server di database e PHP come linguaggio. Ovviamente voi dovrete solo eseguire pochi click, tutte le configurazioni per far funzionare l'ambiente saranno a cura di Xamp
Directory: xampp-1.4.12

DbaMgr2k

Una console amministrativa alternativa per Microsoft MSDE 1.0 e MSDE

Prima di tutto è giusto dire che DbaMgr2k è un progetto completamente italiano. Scritta dall'ottimo Andrea Montanari in VB6, si tratta di un'applicazione che consente di gestire comodamente da un'interfaccia grafica database di tipo MSDE. L'interfaccia funziona sia su MSDE 1.0 che su Sql Server 2000 e persino su Sql Server 7.0. Si tratta di un prodotto molto interessante senza dubbio da tenere nel cassetto degli utilissimi
Directory: \dbamgr2k

Ogre SDK

Un rendering engine emergente

Di motori di rendering 3D ci occupiamo spesso in ioProgrammo.

Quello che vi presentiamo in questo numero ha già 4 anni di vita alle spalle, ma solo da poco ha assunto le proporzioni di un progetto stabile. Questo non vuol dire che il progetto sia poco seguito, al contrario testimonia la serietà di un gruppo che non ha pubblicizzare in precedenza il rilascio di versioni in via di sviluppo e si è concentrato invece sulla produzione di un tool efficiente, affidabile e dotato di tutte le caratteristiche essenziali per essere un grande prodotto.

Il motore è stato ideato per rendere semplici tutte le operazioni di Rendering, si integra perfettamente sia con OpenGL che con Direct3D, supporta sia GCC che Visual C++ e persino Visual C++.NET. Da provare!
Directory: OgreSdk

Un semplice programma con Glade

Un editor di interfaccie visuali

Abbiamo già utilizzato Glade in numeri precedenti di ioProgrammo, in particolare per realizzare l'interfaccia di frontend di un programma Python. Glade è appunto questo, un generatore di interfacce. Potete provare a disegnare per esempio un'interfaccia con Glade e salvare il codice per DevC++ e GTK. Glade non è un compilatore, non genera direttamente l'eseguibile, ma potete usarlo per generare il codice corrispondente all'interfaccia e poi creare l'eseguibile con il compilatore che amate di più.
Directory: GladeWin32

1 **1) LA PARTENZA - Si parte da un nuovo progetto, avendo cura di utilizzare un componente "Window" come contenitore per l'applicazione**

2 **2) L'INTERFACCIA - Posizioniamo gli elementi dell'interfaccia nella window. E' possibile ottenere un posizionamento preciso usando i frame**

3 **3) IL CODICE - Generiamo il codice, scegliendo le opzioni più opportune e infine scegliendo uno dei linguaggi supportati da Glade**

Programmazione concorrente

La maggior parte dei moderni sistemi operativi gestiscono il multitask e il multithread. Alla base di questa utile e potente funzionalità vi è la programmazione concorrente

Il pieno sfruttamento delle risorse hardware di un elaboratore è una finalità primaria nella programmazione dei sistemi operativi. Un generico programma usualmente utilizza più risorse, quali: CPU, memorie primarie, memorie secondarie e dispositivi di I/O. Se tali mansioni vengono svolte in sequenza sarà necessaria una certa quantità di tempo per effettuare l'intera esecuzione. Tale tempo diminuisce, a volte anche sensibilmente, se alcuni compiti si riescono a svolgere in parallelo. Non solo, anche l'attività rispetto ad una singola risorsa, come la RAM, ad esempio per assegnazioni di variabili, può essere a volte compiuta in parallelo. Esistono alcuni costrutti che mettono in pratica la programmazione parallela. Intorno ad essi è stata prodotta una vera e propria teoria. Questa è in continuo fermento. A riprova di ciò, si possono osservare recenti studi che hanno aggiunto nuovi contributi alla realizzazione della programmazione concorrente. Partiremo dalla teoria classica e esamineremo i costrutti primitivi per l'attuazione di questa importante idea.

PER COMINCIARE, UN ESEMPIO

Supponiamo di avere quattro semplici statement di assegnazione all'interno di un generico programma.

$x := d + q + 5$
$y := w - 7$
$z := x + y$
$k := z + 1$

Notiamo subito che alcune istruzioni possono essere svolte in parallelo. Altre vincolano la propria esecuzione allo svolgimento di precedenti assegnazioni. In particolare, le prime due assegnazioni potrebbero essere svolte in contemporanea, mentre la terza dipende dalla prima e dalla seconda. Ancora, la quarta assegnazione può essere svolta solo dopo la terza. Si deduce, che qualora fosse possibile svolgere operazioni in parallelo, ciò condurrebbe ad un sostanziale risparmio di tempo. Con riferimento all'esempio, la contemporanea esecuzione delle prime due assegnazioni segnerebbe un risparmio del 25% rispetto ad un'esecuzione sequenziale di tutte le istruzioni. I vantaggi aumentano sensibilmente se si manipolano variabili strutturate come array.

IL GRAFO DELLE PRECEDENZE

Passo successivo è trovare un modo per rappresentare l'ordine di esecuzione di vari statement. Il grafo delle precedenze, per le caratteristiche di essenzialità e chiarezza, si presta opportunamente al compito. Per comprenderne il suo funzionamento analizziamo un nuovo esempio. In **Figura 1** è riportato un grafo di precedenze che consta di sette statement. Dall'analisi del grafo si rilevano alcuni elementi:

- **S2** e **S3** possono essere eseguite dopo che **S1** sia completata;

Fig. 1: Grafo di precedenze per un programma con sette istruzioni

- **S5** e **S6** possono essere eseguite dopo che **S2** sia completata;
- **S4** può essere eseguita dopo che **S3** sia completata;
- **S7** può essere eseguita dopo che **S4, S5** e **S6** siano completate.

Ovviamente si notano anche altri aspetti, come la contemporaneità di alcune esecuzioni. Ad esempio, *S3* può essere eseguita simultaneamente a *S2, S5* e *S6*. In definitiva si tratta di un grafo aciclico in cui i nodi corrispondono a statement (istruzioni) e gli archi indicano le precedenze. L'istruzione puntata dall'arco esprime la dipendenza di esecuzione, dell'istruzione stessa, dallo statement da cui proviene la freccia.

CONDIZIONI DI CONCORRENZA

Entriamo nei particolari. Individuiamo le condizioni affinché due o più istruzioni possano essere eseguite in modo concorrente. Al tal fine definiamo due insiemi di lettura R(Si) e di scrittura W(Si); anche detti di input e output.

```
R(Si) = { a1, a2, a3, … , am }
W(Si) = { b1, b2, b3, … , bn }
```

Fanno parte del primo insieme le variabili di lettura riferite allo statement S1. All'insieme di scrittura appartengono le n variabili di output. Consideriamo una istruzione per capire la composizione dei due insiemi.

$z := x + y$

dove i valori delle due variabili x e y sono utilizzati per computare il nuovo valore della variabile z. Si noti che il "vecchio" valore di z non è usato nell'assegnazione; ma si ottiene solo un "nuovo" valore come risultato dell'istruzione. Analizziamo i due insiemi per tale istruzione.

```
R(z := x + y) = { x, y }
W(z := x + y) = { z }
```

Facciamo altri esempi per capire meglio la costruzione dei due insiemi di input e output.

```
R(z := z + y + q - 3) = { z, y, q }
W(z := z + y + q - 3) = { z }
R(read(x)) = {  }
W(read(x)) = { x }
```

Si nota che rispetto a procedure di input come la read, cin in C++ per intenderci, l'insieme di lettura è vuoto, mentre quello di scrittura è appunto la variabile letta. In definitiva, potremmo dire che appartengono all'insieme di lettura o input le variabili che vengono in seguito allo statement consultate, è il caso del valore sinistro di una assegnazione. In letteratura informatica questo valore è conosciuto come l-value, mentre il destro r-value. L'insieme di scrittura o output è definito da quelle variabili che in seguito all'istruzione cambiano valore, come nel caso del r-value di una assegnazione o per procedure di input. Due istruzioni S1 e S2 sono eseguibili in modo concorrente se seguono le condizioni di Bernstein:

```
1. R(S1) intersezione W(S2) = {}
2. W(S1) intersezione R(S2) = {}
3. W(S1) intersezione W(S2) = {}
```

La comprensione di tali condizioni è alquanto immediata. Se una variabile è in fase di lettura non si deve simultaneamente trovare in fase di scrittura in un'altra istruzione; e viceversa. Ciò equivale a dire che gli insiemi associati alle intersezioni delle due attività: input e output siano nulli. Inoltre, una variabile non può essere modificata concorrentemente da più istruzioni, come stabilisce l'ultima condizione. Ritornando all'esempio iniziale possiamo verificare come le prime due istruzioni, che chiameremo S1 e S2 possano essere eseguite in modo concorrente. Gli insiemi associati agli statement sono:

```
R(S1) = { d, q }
R(S2) = { w }
W(S1) = { x }
W(S2) = { y }
```

Da cui si può facilmente verificare che le intersezioni di Bernstein sono tutte rigorosamente insiemi nulli. Invece, S3 non può essere eseguita simultaneamente a S1, ma anche a S2, perché vi è un intersezione non nulla, eccola:

```
W(S1) intersezione R(S2) = { x }
```

IL COSTRUTTO FORK E JOIN

È adesso chiaro, almeno spero, il metodo formale per individuare statement che possano essere eseguiti in modo concorrente. È necessario adesso definire un costrutto lessicale per poter attuare la concorrenza. Partiamo dalle radici esaminando le proposizioni che storicamente hanno dato luogo alle teorie sulla concorrenza. Seguire lo stesso percorso tracciato dagli sviluppatori della concorrenza è secondo me il metodo migliore per comprendere appieno l'argomento. La coppia di istruzioni per l'attuazione della concorrenza sono fork e join,

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

introdotte da Conway (una vecchia conoscenza per i lettori di soluzioni), Dennis e Van Horn. L'istruzione fork produce nel programma due esecuzioni concorrenti. La prima esecuzione parte dopo il label descritto di seguito alla parola chiave fork, nell'esempio L. La seconda è l'istruzione appena successiva a fork. Ecco la sintassi:

```
    S1:
    fork L;
    S2;
    …
L : S3
```

Ovviamente, ciò significa che una volta terminata l'esecuzione di S1 partono in contemporanea le esecuzioni delle due istruzioni S2 e S3. Si può associare un significato grafico all'istruzione con riferimento al grafo delle precedenze, espresso in **Figura 2 A**. Si tratta come previsto della biforcazione.

Fig. 2: ***Grafo delle precedenze per il costrutti fork (A) e join (B)***

La fork produce solo due istruzioni che verranno eseguite in modo concorrente, in altri termini solo una biforcazioni con due rami. La join ha il compito di ricombinare due o più esecuzioni concorrenti in un'unica. Ognuna delle esecuzioni concorrenti dovrà richiede di essere congiunta, ovvero fare esplicita istanza di join. Come per la fork all'istruzione è associato un significato per il grafo delle precedenze (**Figura 2 B**). Poiché i tempi di esecuzione potrebbero essere diversi, mentre, una o più esecuzioni sono congiunte, quindi terminate, l'ultima è ancora allocata, e deve terminare la propria esecuzione. Da ciò sorge la necessità di mantenere con un contatore il numero di esecuzioni che hanno fatto join in modo da terminarle tutte tranne l'ultima. Nella join è presente quindi il parametro cont. L'esecuzione della join ha il seguente effetto:

```
cont := cont – 1;
if cont <> 0 then quit;
```

Esaminiamo un semplice e completo esempio che fa uso del costrutto fork e join.

```
(* concorrente *)
S0;
cont := 2;
        fork L1;
        S1;
        Goto L2;
L1:     S2;
L2:     join cont;
        S3:
```

Tale frammento di codice si riferisce al più semplice dei programmi concorrenti, espresso dal grafo delle precedenze di **Figura 3.**

Fig. 3: ***Grafo delle precedenze riferito al codice concorrente***

Ritorniamo al nostro primo esempio. La sequenza di quattro assegnazioni di semplici espressioni aritmetiche. Avevamo stabilito che era possibile eseguire le prime due istruzioni in modo concorrente. Ecco quindi, come si trasforma quel frammento di codice con programmazione concorrente.

```
(* espressioni  concorrenti *)
cont := 2;
        fork L1;
        x := d + q + 5;
goto L2
L1:     y := w - 7
L2:     join cont;
z := x + y
k := z + 1
```

Ritorniamo al grafo delle precedenze di figura 1, sviluppiamolo mediante fork e join. Da notare la join multipla che si implementa ponendo a 3 il contatore. Si otterrà il seguente codice:

```
(* concorrente con join multipla*)
S1;
```

NOTA

COPIA DI FILE CON COBEGIN E COEND

Lo stesso problema della copia di un file realizzato con il costrutto fork e join con la cobegin e coend vede la seguente soluzione.

```
var   f,ff : file of
             tipo_elemento;
  b, bb : tipo_elemento;
  cont : integer;
begin
  rewrite(ff);
  reset(f);
  read(f, b);
  while not eof(f) do
  begin
    bb := b;
    cobegin
       write(ff, bb);
       read(f, b);
    coend;
  end;
  write(ff, b);
end.
```

DIRAMAZIONE MULTIPLA CON FORK

La fork per come è definita sintatticamente prevede la sola biforcazione; split a soli due esecuzioni concorrenti. Per implementare split ad un numero maggiore di due esecuzioni, basta prevedere un istruzione nulla. Quindi, se da S1 si vogliono derivare le tre istruzioni concorrenti S2, S3 e S4, basta prevedere uno statement nullo e realizzare uno schema come mostrato in figura.

```
cont := 3;
        fork L1;
        S2;
        fork L2;
        S5;
        goto L3;
L2:     S6
        goto L3;
L1:     S3;
        S4;
L3:     join cont;
        S7;
```

Il primo modo per attuare la programmazione concorrente è stato esaminato. Esso è sicuramente intuitivo, ma ha un forte limite: non si adatta alla programmazione strutturata, tipica dei linguaggi moderni al alto livello. L'inevitabile ricorso a istruzioni come goto, non rendono il programma strutturato come auspicabile. Troviamo una nuova soluzione.

UN COSTRUTTO PER LINGUAGGI AD ALTO LIVELLO

Il più semplice, oltre che il primo ad apparire sulla scena, costrutto per realizzare la programmazione concorrente con linguaggi ad alto livello è quello proposto da Dijkstra, un'altra vecchia conoscenza! Si tratta di cobegin .. coend, che si trova in letteratura anche nell'accezione parbegin … parend.
La sua sintassi è immediata.

```
cobegin
        S1, S2, … Sn
coend
```

Ogni singolo statement *Si* incluso all'interno di *cobegin … coend*, verrà eseguito in modo concorrente. Facciamo subito esempi. Consideriamo il grafo delle precedenze di **Figura 4**. Esso corrisponde alla semplice sequenza di codice:

Fig. 4: ***Grafo delle precedenze a cui è associato il codice: concorrenza con cobegin .. coend***

```
(* concorrenza con cobegin … coend *)
S0;
```

```
cobegin
        S1; S2; … Sn;
coend;
Sn+1;
```

L'esempio iniziale con le espressioni aritmetiche di assegnazione con il nuovo costrutto viene facilmente tradotto come:

```
cobegin
        x := d + q + 5;
y := w - 7 ;
coend;
z := x + y;
k := z + 1;
```

Mentre, il codice riferito al grafo delle precedenze di **Figura 1** è il seguente.

```
S1;
cobegin
        S2;
        cobegin
                S5; S6;
        coend;
S3;
S4;
coend;
S7;
```

Dagli esempi si nota la facilità d'uso delle nuove istruzioni e la più naturale propensione al concetto di concorrenza. Questa ultima è forse la conseguenza di un'abitudine, personale, allo sviluppo mediante linguaggi strutturati.

CONCLUSIONI

Il primo passo verso la programmazione concorrente ci ha indicato il percorso da seguire. Abbiamo compreso le basi teoriche ed esaminato i costrutti elementari per una prima realizzazione. Un'ultima considerazione circa la comparazione tra i due metodi visti. Va detto che sebbene più tortuoso e meno bello esteticamente la coppia di istruzioni fork e join è più generale di cobegin e coend, ovvero è utilizzabile in corrispondenza di qualsiasi grafo delle precedenze. Insomma è più potente. In alcune situazioni, a dire il vero rare, la coppia cobegin .. coend non è applicabile. Tali situazioni si possono spesso ricondurre a casi di grafi "trattabili" con la coppia di istruzioni ad alto livello. Inoltre, con l'aggiunta di altri costrutti, come i semafori, che esamineremo in futuro, si possono risolvere tutti i tipi di problemi relativi alla programmazione concorrente. In nostro compito sarà esaminarli nelle prossime puntate.

Fabio Grimaldi

COPIA DI FILE CON FORK E JOIN

Analizziamo un caso più concreto. Si vuole copiare un file sequenziale f su un altro file ff. Sfruttiamo le potenzialità della programmazione concorrente. Usiamo due buffer b e bb indispensabili per l'implementazione concorrente. Si userà una sintassi simile a pascal /modula 2.

```
(* Copia concorrente di
                    file *)
var  f,ff : file of tipo_
                elemento;
   b, bb : tipo_elemento;
    cont : integer;
begin
  rewrite(ff);
  reset(f);
  read(f, b);
  while not eof(f) do
  begin
      cont := 2;
      bb := b;
      fork L1;
      write(ff, bb);
      L1: read(f, b);
      L2: join cont;
    end;
  write(ff, b);
end.
```

tipo_elemento è il tipo del singolo elemento che costituisce il file, lasciandolo non definito il codice risulta più generale. In comportamento concorrente in questa situazione è molto utile poiché attenua i consistenti tempi di lettura e scrittura su memoria di massa.

Richard Stallman e la libertà

di Massimiliano Bigatti

L'ultimo Hacker, fondatore di Free Software Foundation, promotore del software libero e principale fautore della nascita e del successo di Linux, ci mostra in un incontro organizzato da AssoEtica, un modo alternativo di concepire l'organizzazione della moderna società del software

Fig. 1: Richard Stallman prende il the

Siamo nel centro di Milano, sotto una pioggerellina sottile. Alla Casa della Cultura, molti sono in trepida attesa di assistere ad un intervento di Richard M. Stallman, il guru di Unix e fondatore della Free Software Foundation. L'evento è organizzato da Assoetica, un'associazione che impegnata nella "business ethics".
Stallman incomincia il suo intervento parlando del nodo centrale del software libero: il fatto che libera gli utenti, gli garantisce una serie di libertà che non avrebbe con software commerciale.
Con voce calma e scandendo le parole affronta le quattro libertà garantite dal software libero.

La libertà 0 è la possibilità di controllare il proprio computer

Consultare il codice sorgente di un programma consente di verificare che non siano presenti applicazioni maligne, come spyware. Stallman fa esempi di software commerciale che contengono funzioni di spia. Windows XP raccoglie informazioni sui programmi installati dall'utente, e li invia a Microsoft non appena il computer viene connesso ad Internet. Lo stesso fanno Real Player e TiVO, si connettono ad un server centrale e comunicano i filmati, URL, programmi TV che l'utente vede o registra. Altri elementi che limitano la libertà dell'utente presenti in programmi commerciali sono le pubblicitià (adware) o le backdoors, strumenti che consentono a potenziali malintenzionati di controllare il computer dell'ignaro utente. Con il software libero tutti questi elementi possono essere rimossi, perché sono disponibili i sorgenti e soprattutto la libertà di modificarli. Un altro elemento importante è che nel software commerciale il potere è nelle mani del produttore, e non in quelle dell'utente.

La libertà 1 è la possibilità di studiare il codice

Molti utenti non esercitano direttamente questo diritto, ma possono pagare altri per farlo.

La libertà 2 è la possibilità di aiutare il prossimo

Per illustrare questa libertà Stallman fa l'esempio del tipico amico che ci chiede una copia di un programma commerciale, perché lo ritiene utile. A questo punto ci si scontra con un dilemma morale: fare una copia illegale del programma, oppure ubbidire alla licenza, e dire di no al proprio amico?
Il dilemma morale ci pone di fronte a due possibili azioni negative: fare una copia illegale o deludere il nostro amico. A questo punto Stallman cerca di capire quale è il male minore, sostentendo che la copia illegale è un male rivolto allo sviluppatore di software proprietario, il soggetto che per primo ha creato il pro-

Fig. 2: Concentratissimo sul suo portatile

blema, e non verso la comunità. Per questo motivo, il software proprietario non aiuta a creare una società migliore, anzi aiuta il sorgere di questi problemi morali, che vanno verso lo sviluppo di una società peggiore.

La libertà 3 è la possibilità di pubblicare una versione modificata del software

È attraverso questa libertà che il software assumerà la forma necessaria agli utenti, non utile al produttore per fare più soldi. L'evoluzione di un programma diviene dunque uno sviluppo democratico, promosso da decisioni collettive. E visto che il numero di programmatori necessari per fare una modifica è generalmente contenuto, per software con una larga utenza il lavoro di pochi si traduce nel vantaggio per molti.

GNU: UN VEICOLO DI LIBERTÀ

Dopo aver illustrato efficacemente e spesso con humor le libertà principali offerte dal software libero, Stallman è passato ad illustrare la nascita della Free Software Foundation, e del progetto GNU, non senza togliersi qualche sassolino dalla scarpa. Stallman arrivò all'idea di creare un clone del sistema Unix dopo aver preso una serie di decisioni tecniche, di portabilità, e considerando molte altre questioni, come l'accettabilità di un nuovo sistema da parte degli utenti. Oppure la necessità di mantenere le interfacce. Ad un certo punto il progetto GNU portò allo sviluppo di un completo sistema operativo, ma mancava ancora il Kernel. Stallman ammette come l'avvento di Linux abbia dato un impulso considerevole alla diffusione del software libero, sottolineando però che inizialmente il prodotto di Linus Torvalds non era stato rilasciato sotto licenza GPL, non era inizialmente software libero.
Un altro punto caro a Richard Stallman è il nome che comunemente si utilizza per i sistemi operativi basati su Linux; si tende a dimenticare infatti che molte componenti dei sistemi operativi come Red Hat, Suse o Mandrake utilizzano software GNU. RMS dunque propone di chiamarli sistemi GNU/Linux. Il progetto GNU è insomma un veicolo delle idee di libertà di Stallman e della FSF. Stallman conclude il suo intervento discutendo delle varie leggi che minacciano il software libero, e che sono un vero e proprio scandalo ed attentato alla libertà delle persone. Negli Stati Uniti c'è già la DMCA che restringe le aree di applicabilità del software libero: ad esempio non è possibile produrre negli US software libero che permetta di leggere un DVD, perché non è consentito rendere facile all'utente capire i meccanismi di crittazione dei dati.
Nell'Unione Europea abbiamo anche noi i nostri problemi, soprattutto in merito ai brevetti; una direttiva molto negativa è stata bloccata grazie al ministro polacco. E prima che Stallman potesse dire "Thank you, Poland", la platea scoppiava in un applauso.

Fig. 4: Mauro Graziani presidente di AssoEtica

Fig. 3: Decisamente a suo agio in una fase della conferenza

Biblioteca

ON LINE

DOCEBOLMS

Dai creatori di Spaghettibrain e SpaghettiLearning, i primi a portare in Italia CMS che hanno fatto la storia come PHPNuke, arriva questo DoceboLMS, un progetto OpenSource per l'apprendimento a distanza che promette di diventare uno standard in questo settore

http://www.docebolms.com

PROGRAMMAZIONE.IT

Un sito ricco di contenuti, un raccoglitore che mette insieme il meglio della programmazione e lo propone in una forma aggregata. Non ci sono moltissimi contenuti proprietari, ma la raccolta di puntatori e link ad articoli di ottima fattura spesso nascosti nel marasma degli indici ufficiali è altissima

http://www.programmazione.it

NEWBIE.IT

Un portale dedicato a chi non ha moltissima esperienza di informatica ma vuole comunque imparare. Ricco di informazioni, manuali e contenuti proposti in una forma semplice, si presenta come un'ottimo punto di riferimento per chi inizia

http://www.newbie.it

XHTML IN PRATICA

Ancora non molti conoscono a fondo l'XHTML, il nuovo linguaggio nato dalla fusione di HTML e XML. Roberto Abbate, ci conduce per mano nella comprensione di questo nuovo modo di programmare. Il libro è colmo di esempi che consentono a chi legge di essere immediatamente operativo, nonostante questo vengono approfondite anche le motivazioni che hanno indotto il W3C ad apportare modifiche sostanziali al linguaggio. Interessanti i due capitoli su CSS e Javascript. Il primo illustra nel dettaglio i fogli di stile, grazie ai quali è possibile una formattazione della pagina precisa ed estremamente efficace. Il secondo approfondisce la capacità di JavaScript di interagire con l'utente, mettendo a disposizione del programmatore una serie di oggetti ed eventi che consentono di modificare dinamicamente una pagina HTML. Nonostante l'arrivo di nuovi ed evoluti linguaggi, i tre aspetti approfonditi nel libro di Roberto Abbate rimangono la base della buona programmazione web: XHTML, fogli di stile, JavaScript. Interessante anche il metodo di diffusione del libro, acquistabile unicamente online all'indirizzo *http://www.manuali.net/affiliati/scheda.asp?id=731* e al costo di € 25.

Difficoltà: Bassa • Autore: Roberto Abbate • Editore: J-GROUP • Pagine: 256

JAVASCRIPT LA GUIDA

La quinta edizione di un libro che è effettivamente una guida sicura per chi programma in JavaScript. ben 1188 pagine dedicate a chi programma in questo linguaggio. Si va dalle basi fino ad affrontare le classi più importanti passando per i particolari del modello ad oggetti, esplicitando i meccanismi interni con cui funzionano i browser e infine parlando della gestione degli eventi. Non è roba da poco se si considera che avere una buona conoscenza di JavaScript significa essere in grado di risolvere un buon 50% dei problemi che si presentano in fase di programmazione di un'Interfaccia Web. L'autore Danny Goodman è una garanzia in quanto a conoscenza della materia, da molti è infatti indicato come uno dei Guru di JavaScript. In taluni casi il fatto che un autore sia un'eminenza grigia di una materia implica che i lavori risultanti siano sufficientemente teorici, non è questo il caso. Il libro è ricco di esempi, molti dei quali affidati a uno sviluppatore sul campo: Michael Morrison che non ha mancato di produrre elementi in grado di risolvere i problemi reali di chi legge

Difficoltà: Medio-Alta • Autori: Danny Goodman, Michael Morrison • Editore: McGraw Hill • ISBN: 88-386-4406-3 • Anno di pubblicazione: 2004 • Lingua: Italiana • Pagine: 1188 • Prezzo: € 63

SITI WEB CON PHP E MYSQL

Titolo poco fantasioso, ma d'altra parte centra perfettamente l'argomento. Il duo in questione fa girare da solo almeno il 40% del Web. La maggioranza dei prodotti OpenSource oggi in commercio è sviluppata utilizzando PHP e MySQL. Sono molti i vantaggi nell'uso di queste due tecnologie. Prima di tutto un tempo di sviluppo molto ridotto dovuto in massima parte al fatto che PHP dispone di un numero di funzioni native elevatissimo e tale da rendere superfluo scrivere estensioni proprietarie, d'altra parte dove questo si renda necessario il linguaggio supporta sia la programmazione a oggetti che quella procedurale. MySQL per contro è estremamente veloce, dispone di funzionalità tipiche di DB Server commerciali molto costosi ed è altamente integrato con PHP. Il libro spiega nel dettaglio ogni problematica relativa all'interazione fra PHP e MySQL, rivelandosi un'ottima guida per chi inizia come per chi ha già esperienza.

Difficoltà: Medio-bassa • Autori: Kevin Yank • Editore: Mondadori Informatica • ISBN: 88-04-53965-8 • Anno di pubblicazione: 2004 • Lingua: Italiana • Pagine: 308 • Prezzo: $ 30